HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Maik Benndorf

Konzeption und prototypische Implementierung eines OPC-OR-Mappers für die Datenübernahme aus einer Automatisierungsanlage

Mittweida, 2011

Fakultät **Mathematik/Naturwissenschaften/Informatik**

BACHELORARBEIT

Konzeption und prototypische Implementierung eines OPC-OR-Mappers für die Datenübernahme aus einer Automatisierungsanlage

Autor:
Maik Benndorf

Studiengang:
Informatik

Seminargruppe:
IF08w1-B

Erstprüfer:
Prof. Dr.-Ing. Wilfiried Schubert

Zweitprüfer:
Dipl.-Ing. Jan Gebauer

Einreichung:
Mittweida,

Verteidigung/Bewertung:
Mittweida, September 2011

Bibliografische Angaben

Benndorf, Maik: Konzeption und prototypische Implementierung eines OPC-OR-Mappers für die Datenübernahme aus einer Automatisierungsanlage, 133 Seiten, 63 Abbildungen, Hochschule Mittweida, Fakultät Mathematik/Naturwissenschaften/Informatik

Bachelorarbeit, 2011

Referat

Ziel dieser Arbeit ist die Konzeption und prototypische Implementierung eines Frameworks zur Konfiguration und Ausführung von Datenaustauschen zwischen OPC-Servern und Datenbankmanagementsystemen. Zu diesem Framework gehört ein Frontend, das es ermöglicht, die beiden benannten Funktionen, ohne jeglichen Programmieraufwand, zu tätigen. Im ersten Teil der Arbeit werden die verschiedenen am Prozess beteiligten Technologien beleuchtet. Der zweite Teil beschäftigt sich mit dem Entwicklungsprozess. Dieser wird durch die Wahl eines Vorgehensmodells für den Softwareentwicklungsprozess eingeleitet und führt über Pflichtenheft, die objektorientierte Analyse und das objektorientierte Design hin zur Entwicklung des prototypischen Frameworks. Abgeschlossen wird diese Arbeit durch die Tests der Klassenbibliothek und einem Ausblick.

I. Inhaltsverzeichnis

Inhaltsverzeichnis I
Abbildungsverzeichnis II
Tabellenverzeichnis III
Listings IV
Vorwort V
1 Einleitung 1
1.1 Motivation und Zielsetzung 1
1.2 Aufbau der Arbeit 1
2 Aufgabenanalyse 3
2.1 Anforderungen an OR Mapper allgemein 3
2.2 Anforderungen an den OPC-OR-Mapper 3
2.3 Abgrenzung 5
3 Technologische Grundlagen 7
3.1 OPC 7
3.1.1 Der Begriff OPC 7
3.1.2 Die OPC-Spezifikationen 8
3.1.3 Die Kommunikation 10
3.1.4 Der OPC-Server 11
3.1.5 Der OPC-Client 11
3.1.6 Das OPC-Item 11
3.1.7 Die OPC-Group 12
3.2 Datenbanken 13
3.2.1 Der Begriff Datenbanken 13
3.2.2 Relationale Datenbanken 13
3.2.3 Objektrelationale Datenbanken 14
3.2.4 Objektorientierte Datenbanken 16
3.3 OR Mapping 18
3.3.1 Der Begriff OR Mapping 18

3.3.2 Formen des OR Mappings 18
3.3.3 Forward- und Backward Mapping 19
3.3.4 Das Caching 19
3.3.5 Das Fetching 20
3.3.6 Probleme beim OR Mapping 20
3.3.7 LINQ To SQL 21
3.3.8 NHibernate 23
3.3.9 Zusammenfassung 24
4 Das Lösungskonzept 25
4.1 Softwareentwicklungsprozess 25
4.1.1 Vorgehensmodelle 25
4.1.2 Wahl des Vorgehensmodells 27
4.2 Auswahl eines Datenbankmodells 29
4.3 Auswahl eines OR Mappers 30
4.4 Verwendete OPC-Spezfikation 31
4.5 Verwendung einer Datenzugriffsschicht auf OPC-Server 31
4.6 Verwendung einer Datenzugriffschicht auf die Datenbanken 32
4.7 Zusammenfassung 33
5 Die Umsetzung des Prototypen OPC-OR-Mapper 35
5.1 Die Anwendung des Vorgehensmodells während der Entwicklung 35
5.2 Die Beschreibung der Anforderungen in der Analyse durch das Lastenheft 36
5.2.1 Zielbestimmungen 36
5.2.2 Produkteinsatz 36
5.2.3 Produktübersicht 38
5.2.4 Produktfunktionen 39
5.2.5 Produktdaten 42
5.2.6 Produktleistungen 43
5.2.7 Qualitätsanforderungen 43
5.3 Die Umsetzung der Anforderungen in der Definition 43
5.3.1 Das Pflichtenheft 44
5.3.2 Die objektorientierte Analyse 45

5.4 Die Erweiterung der Definition im Entwurf 54
5.4.1 Einflussfaktoren 54
5.4.2 Entscheidung zur Datenhaltung 55
5.4.3 Das Objektorientierte Design 55
5.5 Die technische Realisierung in der Implementierung 59
5.6 Besonderheiten während der Implementierung 60
5.7 Das Testen im Softwarelebenszyklus 64
5.7.1 Einsatz des Testframeworks NUnit 64
5.7.2 Klassentest 64
5.7.3 Modultest 65
5.7.4 Integrationstest 65
5.7.5 Systemtest 66
5.7.6 Abnahmetest 66
5.7.7 Testfälle 67
5.7.8 Testumgebung 67
5.7.9 Testergebnisse 68
6 Zusammenfassung und Ausblick 71
6.1 Ergebnisse 71
6.2 Reflektion des Erreichten 72
6.3 Ausblick 73
6.3.1 Möglichkeiten der Weiterentwicklung 73
6.3.2 Praktischer Einsatz 73
Literaturverzeichnis 75
A Das Pflichtenheft 79
A.1 Zielbestimmungen 79
A.1.1 Musskriterien 79
A.1.2 Wunschkriterien 80
A.1.3 Abgrenzungskriterien 80
A.2 Produkteinsatz 81
A.2.1 Anwendungsbereiche 81
A.2.2 Zielgruppen 81

A.2.3 Betriebsbedingungen 81
A.3 Produktübersicht 82
A.3.1 Verbindung zur Datenbank verwalten 83
A.3.2 Verbindung zur OPC-Server verwalten 83
A.3.3 Konfiguration verwalten 84
A.3.4 Verwaltung des Datentransfers 84
A.3.5 Datenflussdiagramm 85
A.4 Produktfunktionen 85
A.5 Produktdaten 90
A.6 Produktleistungen 91
A.7 Qualitätsanforderungen 91
A.8 Benutzeroberfläche 92
A.9 Nichtfunktionale Anforderungen 94
A.10 Technische Produktumgebung 94
A.10.1 Software 94
A.10.2 Hardware 94
A.10.3 Orgware 95
A.10.4 Produktschnittstellen 95
A.11 Spezielle Anforderungen an die Entwicklungsumgebung 95
A.11.1 Software 95
A.11.2 Hardware 95
A.11.3 Orgware 95
A.11.4 Entwicklungs-Schnittstellen 95
A.12 Gliederung in Teilprodukte 96
A.12.1 Framework 96
A.12.2 Programmoberfläche 96
B Der Grobentwurf 97
B.1 Einflussfaktoren 97
B.2 UseCase Beschreibungen 97
B.2.1 Verbindung zur Datenbankverwalten 97
B.2.2 Verbindung zum OPC-Server verwalten 101

B.2.3 Konfiguration verwalten 105
B.2.4 Verwaltung des Datentranfers 108
B.3 Packagediagramm 110
B.3.1 Diagramm 111
B.3.2 Anmerkungen 111
B.4 Klassendiagramm 112
B.4.1 Connectoren 112
B.4.2 Konfiguration 114
B.4.3 Datentransfer 115
B.4.4 Anwendung 115
B.5 Sequenzdiagramm 116
C Der Feinentwurf 119
C.1 Einleitung 119
C.2 Die Klassendiagramme 119
C.3 Die Aktivitätsdiagramme 121
C.3.1 Datenbankverbindung verwalten 123
C.3.2 Opc-Verbindung verwalten 124
C.3.3 Konfiguration verwalten 125
C.3.4 Datentransfer verwalten 126
C.4 Die Sequenzdiagramme 126
C.4.1 Datenbank-Verbindung anlegen 127
C.4.2 Opc-Verbindung anlegen 127
C.4.3 Konfiguration verwalten 128
C.4.4 Datentransfer verwalten 130
D Das Data Dictionary 131
Erklärung 133

II. Abbildungsverzeichnis

2.1 UseCase-Diagramm OPC-OR-Mapper 4

3.1 OPC-Client-Server-Struktur 7

3.2 Das objekt-relationale Mapping 18

3.3 Das Fetching 20

4.1 Das angepasste V-Modell 29

4.2 Matrix zur Wahl des Datenbanksystems 29

4.3 Umweltdiagramm 33

5.1 Das angepasste V-Modell 35

5.2 UseCase Produktübersicht 38

5.3 Die Gesamtarchitektur als Packagediagramm 46

5.4 Die OpcConnection in der objektorientierten Analyse 47

5.5 Die Connectoren in der objektorientierten Analyse 48

5.6 Die Configuration in der objektorientierten Analyse 48

5.7 Der Settingsmanager in der objektorientierten Analyse 49

5.8 Das LinkItem und das LinkItemSet in der objektorientierten Analyse 49

5.9 Der LinkItemManager in der objektorientierten Analyse 50

5.10 Das Paket Configuration in der objektorientierten Analyse 50

5.11 Das Converter in der objektorientierten Analyse 51

5.12 Das Paket DataTransfer in der objektorientierten Analyse 52

5.13 Das Frontend in der objektorientierten Analyse 53

5.14 Klassendiagramm der Konfiguration 56

5.15 Klassendiagramm der Konfiguration Globals 57

5.16 Klassendiagramm des Datentransfers 58

5.17 Das DataTypeDataSet 60

5.18 Das DataTypeDataSet in der Datenraster-Ansicht 61

6.1 Das Hauptfenster des OPC-OR-Mapper 71

6.2 Der OPC-OR-Mapper in Aktion 72

A.1 Erweitertes UseCase Produktübersicht 82

A.2 Erweitertes UseCase Verbindung zur Datenbank verwalten 83
A.3 Erweitertes UseCase Verbindung zum OPC-Server verwalten 83
A.4 Erweitertes UseCase Konfiguration verwalten 84
A.5 Erweitertes UseCase Verwaltung des Datentransfers 84
A.6 Datenflussdiagramm 85
A.7 Die Oberflächenstudie der Konfiguration 92
A.8 Die Oberflächenstudie des Datentransfers 92
A.9 Der Datenbankdialog 93
A.10 Der OPC-Server-Dialog 93
A.11 Der Konfigurationsdialog 94
B.1 Verbindung zur Datenbank verwalten 97
B.2 Verbindung zum OPC-Server verwalten 101
B.3 Konfiguration verwalten 105
B.4 Verwaltung des Datentransfers 108
B.5 Das PackageDiagramm 111
B.6 Klassendiagramm DbConnector 113
B.7 Das Paket Configuration in der objektorientierten Analyse 114
B.8 Das Paket DataTransfer in der objektorientierten Analyse 115
B.9 Das Frontend in der objektorientierten Analyse 116
B.10 Sequenzdiagramm Anlegen einer Konfiguration 117
C.1 Der OpcConnector im objektorientierten Design 120
C.2 Der DbConnector im objektorientierten Design 120
C.3 Das GUI im objektorientierten Design 121
C.4 Das Aktivitätsdiagramm für die Gesamtübersicht 122
C.5 Das Aktivitätsdiagramm Datenbankverbindung verwalten 123
C.6 Das Aktivitätsdiagramm Datenbankverbindung verwalten 124
C.7 Das Aktivitätsdiagramm Datenbankverbindung verwalten 125
C.8 Das Aktivitätsdiagramm Datenbankverbindung verwalten 126
C.9 Das Sequenzdiagramm Datenbank-Verbindung anlegen 127
C.10 Das Sequenzdiagramm Opc-Verbindung anlegen 127
C.11 Das Sequenzdiagramm Konfiguration anlegen 128

C.12 Das Sequenzdiagramm Konfiguration ändern 129
C.13 Das Sequenzdiagramm Konfiguration löschen 129
C.14 Das Sequenzdiagramm Datentransfer starten 130
C.15 Das Sequenzdiagramm Datentransfer stoppen 130

III. Tabellenverzeichnis

3.1 Vergleich der OR Mapper 24
5.1 Produktüberischt 40
5.2 Produktdaten 43
5.3 Produktleistungen 43
5.4 Produktleistungen 43
5.5 Testumgebung Rechner 1 68
5.6 Testumgebung Rechner 2 68
5.7 Der Testfall Datenbankverbindung anlegen 69
A.1 Pflichtenheft Revisionstabelle 79
A.2 Produktüberischt 88
A.3 Produktdaten 90
A.4 Produktleistungen 91
A.5 Erweiterte Qualitätsanforderungen 91
B.1 Grobentwurf Revisionstabelle 97
B.2 UseCase-Beschreibung Datenbankverbindung herstellen 98
B.3 UseCase-Beschreibung Datenbankverbindung ändern 99
B.4 UseCase-Beschreibung Datenbankverbindung löschen 100
B.5 UseCase-Beschreibung Verbindung zum OPC-Server herstellen 102
B.6 UseCase-Beschreibung Verbindung zum OPC-Server ändern 103
B.7 UseCase-Beschreibung Verbindung zum OPC-Server löschen 104
B.8 UseCase-Beschreibung Konfiguration erstellen 106
B.9 UseCase-Beschreibung Konfiguration ändern 107
B.10 UseCase-Beschreibung Verbindung zum OPC-Server löschen 108
B.11 UseCase-Beschreibung Datentransfer starten 109
B.12 UseCase-Beschreibung Datentransfer stoppen 110
B.13 Anmerkungen Packagediagramm 112
C.1 Feinentwurf Revisionstabelle 119
D.1 DataDictionary Opc-Server-Verbindung 131

D.2 DataDictionary Datenbank-Verbindung 131
D.3 DataDictionary LinkItem 131
D.4 DataDictionary LinkItemSet 132
D.5 DataDictionary Konfiguration 132

IV. Listings

3.1 Benutzerdefinierte Datentypen 14
3.2 Vererbung 15
3.3 Tabelle vom benutzerdefinierten Datentyp 15
3.4 Punktnotation im objektrelationalen Modell 15
3.5 Typkonstruktoren 16
3.6 Abfrage unter SQL 22
3.7 Abfrage unter C-Sharp 22
3.8 Abfrage unter LINQ 22
4.1 ExecuteNonQuery unter dem Framework 32
4.2 ExecuteNonQuery unter der Datenzugriffsschicht 32
5.1 Die MatchDataTypes-Methode 61
5.2 Die Methode um das LinkItem zu clonen 62
5.3 Die Methode um das LinkItemSet zu clonen 62

V. Vorwort

Diese Arbeit zum Thema „Konzeption und prototypische Implementierung eines OPC-OR-Mappers für die Datenübernahme aus einer Automatisierungsanlage“ entstand im Rahmen des Studiums an der Hochschule Mittweida (FH) im Studiengang Informatik und in Zusammenarbeit mit der HGDS Hoffbauer und Gebauer Datenservice GmbH aus Riesa.

1 Einleitung

1.1 Motivation und Zielsetzung

In der heutigen Industrie und Wirtschaft geht es vor allem um den effizienten Einsatz von Resourcen, dazu zählen Kapital, Mitarbeiter und auch Maschinen um somit Geschäftsprozesse zu optimieren. Moderne ERP-Systeme (Enterprise Resource Planing) beschreiben diese Aufgabe. Der Optimierung geht dabei zunächst die Erfassung und Analyse der aktuellen Situation voraus. Gerade im Hinblick auf den Einsatz von Maschinen, müssen deren Daten zunächst erfasst und analysierbar gemacht werden. Automatisierungsanlagen steuern nicht nur die Abläufe der Maschinen sondern liefern auch Werte. Diese Werte müssen für den Einsatz moderner ERP-System in Datenbanken abgebildet werden. Es fehlt also eine Schnittstelle zwischen den Objekten der Automatisierungsanlage und den Datenbanken.

Das Stichwort dazu heißt OR (Objekt Relationales) Mapping. Ziel dieser Arbeit ist die Konzipierung und Implementierung eines Frameworks , das Funktionen bereitstellt die von der Erstellung der Verbindungen zu Automatisierungsanlagen (Quelle) und Datenbanken(Ziel) über die Verknüpfung von den Quell- und den Zieldaten bis hin zur Ausführung von Datenaustauschen auf der Grundlage der angelegten Verbindungen und Verknüpfungen umfasst. Diesem Paket ist ein Frontend hinzuzufügen, das es ermöglicht die zuvor beschrieben Funktionen des Frameworks, ohne jeglichen Programmieraufwand, zu nutzen.

1.2 Aufbau der Arbeit

Im ersten Kapitel werden zunächst die Aufgaben an OR-Mapper im allgemeinen formuliert, um daraus die Anforderungen an das zu erstellende Framework abzuleiten und diese schließlich abzugrenzen. Das zweite Kapitel dient dazu, die verwendeten Technologien zu beleuchten und Unterschiede aufzuzeigen, dabei wird abschließend auch auf sich bereits am Markt befindende OR-Mapper eingegangen. Das Lösungskonzept (Kapitel 3) befasst sich zunächst mit den Vorgehensmodellen um dann im Rückblick auf das Kapitel Technologien eine Auswahl zu treffen, die im letzten Kapitel, der Umsetzung, ihre Anwendung finden. Abgeschlossen wird diese Arbeit von einer Reflektion und einem Ausblick.

2 Aufgabenanalyse

2.1 Anforderungen an OR Mapper allgemein

Unter dem OR-Mapping versteht man das Abbilden von Objekten der objektorientierten Programmierung in Tabellen relationaler Datenbanken. Eine Anwendung die sich OR-Mapper nennt sollte laut Gavin King und Christian Bauer[1] die folgenden Punkte beachten:

1. Eine API zur Durchführung einfacher CRUD(Create, Read, Update, Delete) -Operationen mit Objekten persistenter Klassen.
2. Eine Sprache oder API für die Formulierung von Abfragen, die sich auf Klassen und Klasseneigenschaften beziehen.
3. Eine Einrichtung für die Spezifizierung des Mappings von Metadaten.
4. Eine Technik, um mit transaktionalen Objekten zu interagieren, wie Dirty Checking, Lazy Association Fetching.

2.2 Anforderungen an den OPC-OR-Mapper

Der OPC-OR-Mapper ist in erster Linie ein Framework, das einen lesenden Zugriff auf OPC-Server und schreibenden Zugriff auf die Datenbanken bietet. Um diese Anforderungen zu gewärleisten, müssen die folgenden Funktionalitäten erfüllt werden:

- die Verwaltung der Datenbankverbindungen,
- die Verwaltung der OPC-Server-Verbindungen,
- die Verwaltung der Konfigurationen und
- die Verwaltung der Datentransfer.

Daraus ergibt sich das in der Abbildung 2.1 dargestellte UseCase-Diagramm.

[1] vergleiche [KiBa07]

Abbildung 2.1: UseCase-Diagramm OPC-OR-Mapper

Unter der Verwaltung der Datenbankverbindung, OPC-Server-Verbindung und Konfigurationen ist das Hinzufügen, Ändern und Löschen einer solchen zu verstehen. Die Verwaltung der Konfiguration ist dahingehend zu erweitern, als dass auch die Verwaltung der abzubildenden Daten integriert sein muss. Die Verwaltung der Daten beinhaltet, das Anlegen eines neuen Datensatzes und das Zuweisen der dazugehörigen Zielfunktion, sowie das Ändern und Löschen dieser. Ein Datensatz ist dabei als Objekt des OPC-Servers zu verstehen (z.B. der Temperaturwert einer Automatisierungsanlage).

Dem Zuweisen einer Zielfunktion muss eine Prüfung der Verträglichkeit der Datentypen vorausgehen.

Das Starten und Stoppen der zuvor angelegten Konfigurationen als Datentransfer ist in der Verwaltung der Datentransfer zu finden.

Es gilt weiterhin zu beachten, dass Transaktionen vollständig ausgeführt werden müssen, dass heißt ein Lesen der Werte vom OPC-Server und das nicht erfolgte Schreiben in eine Datenbank ist zu vermeiden, da diese Werte nicht wieder zur Verfügung stehen und somit verloren gehen.

Um die Datenintegriät zu wahren, darf das Löschen von Verbindungen, die in Verwendung sind, nicht möglich sein.

2.3 Abgrenzung

In der prototypischen Implementierung, wird die Software nicht in der Lage sein, die folgenden Funktionalitäten zu erfüllen:

- Es wird davon ausgegangen, dass die Zielfunktionen bzw. Tabellen schon existieren. Somit ist es nicht möglich Daten in ein nicht existentes Ziel zu mappen bzw. dieses mit Hilfe des Programms anzulegen.
- Bisher ist nur ein lesender Zugriff auf die Daten des OPC-Servers sowie schreibender Zugriff auf die Daten in der Datenbank angedacht. Es gilt jedoch zu beachten, dass die triggerbasierte Aktualisierung einen schreibenden Zugriff erfordert. Der schreibende Zugriff ist notwendig, weil bei dieser Methode das Feld „SignalsValid“ nach dem Lesen gesetzt werden muss, um das Fortschreiben des OPC-Servers zu ermöglichen. Andernfalls ist nur das Auslesen eines einzigen Datensatzes möglich.
- Die Auswahl der Datenbankprovider beschränkt sich in der prototypischen Implementierung auf den Oracle und den SQL-Server.
- Die im Prototyp unterstützten OPC-Server sind der SimaticNet von Siemens und der INAT OPC-Server.
- Die zu mappenden Daten des OPC-Servers müssen händisch erfasst und der dazugehörige Datentyp ausgewählt werden.

3 Technologische Grundlagen

3.1 OPC

3.1.1 Der Begriff OPC

„OPC ist die Abkürzung für 'OLE for Process Control' wobei OLE heute restrukturiert und in ActiveX umbenannt wurde. Bei OPC handelt es sich um einen industriellen Standard. In diesem Standard werden Methoden definiert, welche den Datenaustausch zwischen Automatisierungsanlagen und Clients mit Microsofts Betriebssystemen ermöglichen. Für die Kommunikation zwischen den Anwendungen wird derzeit hauptsächlich die Schnittstelle DCOM von Microsoft verwendet.“ [2]

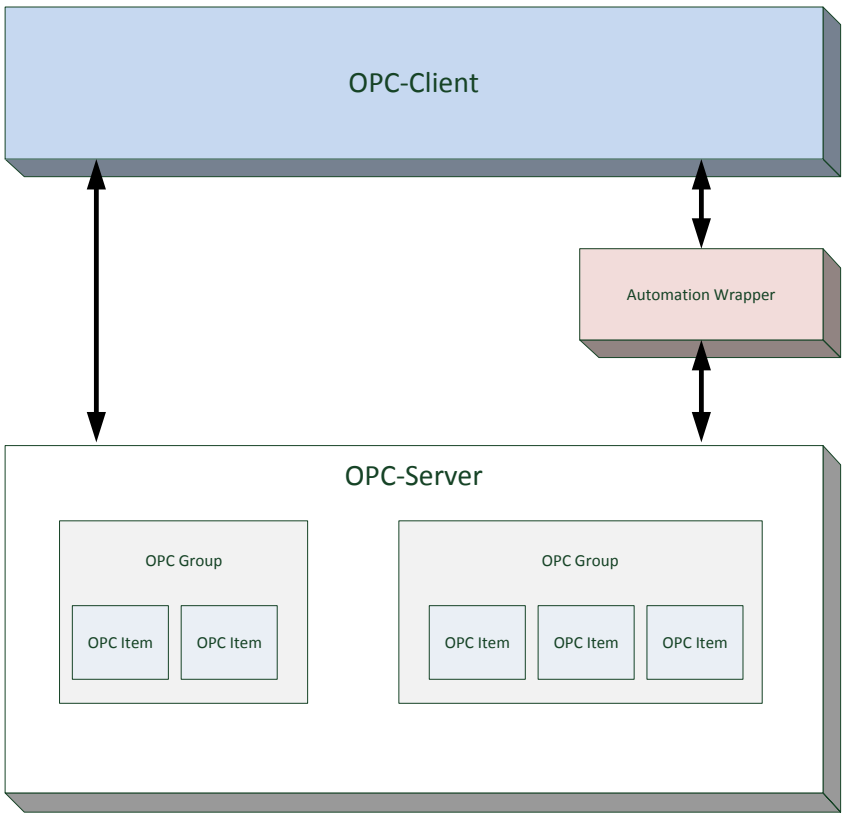

Abbildung 3.1: OPC-Client-Server-Struktur

[2] Vgl.: [MaBe11] Seite 17

3.1.2 Die OPC-Spezifikationen

OPC DA

Der „Data Access(DA)“ ist die bekannteste Spezifikation und wird für die Verarbeitung von Real-Time-Daten eingesetzt. Dabei sind die drei Eigenschaften:

- value,
- quality of the value und
- timestamp

mit OPC DA verbunden. Die Spezifikation gibt dabei an, dass alle drei Attribute an den Client übermittelt werden müssen. OPC DA baut beim Datentransport auf Microsofts DCOM-Technologie.

OPC HDA

Während OPC Data Access den Zugriff auf Real-Time-Daten, die sich fortlaufend ändern, ermöglicht, wird vom OPC Historical Access der Zugriff auf archivierte Prozessdaten unterstützt.

Auch HDA verwendet wie schon DA die DCOM Technologie, zusätzlich werden die Security-Features von DCOM, wie z.B. User- Authentifizierung, Berechtigungen oder auch den Verschlüsselungsservice, verwendet. [3]

OPC A/E

Die Spezifikation OPC Alarms und Events ist für Automatisierungsanlagen gedacht, die Alarme und Events erzeugen. Dabei werden die Alarme und Events vom OPC-A/E-Server nicht erzeugt sondern lediglich, für den konfigurierten Speicherbereich der Automatisierungsanlage, weitergemeldet.

OPC DX

In der Spezifikation OPC Data eXchange wird definiert, auf welche Weise OPC-Server untereinander Daten austauschen. Bei OPC DX handelt es sich nicht um eine neue Art des Datenaustauschs, stattdessen baut sie auf die Möglichkeiten von OPC DA, sie definiert das Verhalten des OPC DX-Servers, indem sie sich auf die Steuerung und das Überwachen des Datentransfers von DA-Servern zu sich selbst bezieht.[4]

[3] Vgl.: [JInt10]
[4] Vgl.: [Advo11]

OPC XML DA

OPC XML Data Access ist die Spezifikation zur XML- basierten Übertragung von Real-Time-Daten. Da zur Zeit der Realisierung diese Spezifikation klar war, dass es eine DCOM-unabhängige Spezifikation in Form von OPC UA geben wird, fand OPC XML DA nur wenig Verwendung.

OPC UA

Nachdem COM/DCOM in erheblichem Maß an der Verbreitung von OPC beteiligt waren, waren doch die Probleme, die sich daraus ergaben nicht weg zu diskutieren. Zu den Problemen gehörten:

- die Konfigurationsprobleme von DCOM,
- dass die Timeouts sich nicht konfigurieren ließen,
- die Abhängigkeit von den Windows-Betriebssystemen durch COM bzw. DCOM,
- das von COM/DCOM keine Sicherheit geboten wird,
- das um DCOM nutzen zu können, Firewalls geöffnet werden müssen,
- und schließlich, dass COM bzw. DCOM als Blackbox anzusehen sind und die Entwickler somit den Fehlern ausgeliefert sind.[5]

Um den derzeitigen und auch zukünftigen Anforderungen Stand zu halten, setzt die OPC-Foundation mit dieser neuen Spezifikation, der „Unified Architecture“, auf die serviceorientierte Architektur (SOA).
Dieser neue Standard ist vor allem plattformunabhängig und findet somit Anwendung in Kleinstgeräten bis hin zu Mainframes. Das der Stack sowohl für Multithreaded-Betrieb als auch für den Singlethreaded/Singletask-Betrieb kompiliert werden kann, ist ein großer Vorteil der Serviceorientierten Architektur.
Weitere wichtige Punkte sind die portable ANSI-C-Implementierung, die Skalierbarkeit von Embedded Controllern bis hin zu Mainframes, die Konfiguration der Timeouts für alle Services, das Chunkink von großen Datenpaketen und auch die Performance und Sicherheit. OPC UA bietet dabei Sicherheitsmeachanismen wie Verschlüsselung, Autorisierung und Authentifizierung.
Nach mehr als drei Jahren Spezifikationsarbeit und einem Jahr Implementierung eines Prototypen wurde im Herbst 2006 die erste Version veröffentlicht.[6]

Mit der Einführung von OPC-UA wurde das Konzept von OPC grundlegend überarbeitet. Die OPC-UA-Architektur ist in mehrere logische Ebenen aufgeteilt. Alle von OPC definierten Services sind abstrakte Methodenbeschreibungen, sind protokollunabhän-

[5] Vgl.:[Asco11]
[6] Vgl.: [Asco11]

gig und bilden die Basis für die gesamte OPC-UA-Funktionalität. Die Transportschicht setzt diese Methoden in einem Protokoll um, d. h. es serialisiert/deserialisiert die Daten und sendet diese über das Netzwerk. Momentan werden zwei Protokolle verwendet, ein binäres und ein auf Webservice basierendes Protokoll. Es besteht aber auch die Möglichkeit, bei Bedarf weitere Protokolle zu ergänzen. Das bekannte hierarchische OPC-Informationsmodell weicht einem sogenannten „Full-Mesh-Network“ aus sogenannten Node's. Ein Node ist vergleichbar mit einem XML-Tag. Jeder Node kann Attribute besitzen, die gelesen werden können und wird sowohl für Nutzerdaten als auch für alle Arten von Metadaten verwendet.[7] Weitere wichtige Features von OPC UA sind:

- Unterstützung durch die Verwendung von Redundanzen,
- die Verbindungsüberwachung in beide Richtungen, d. h. sowohl Server als auch Client bemerken die Unterbrechung,
- Daten werden gepuffert, dass bedeutet im Falle einer Verbindungsunterbrechungen können die Daten nochmals angefordert werden und sind nicht verloren.

3.1.3 Die Kommunikation

Die Kommunikation kann in den folgenden drei Arten unterschieden werden.

1. COM/DCOM
2. XML Webservice
3. OPC UA

Unter COM bzw. DCOM etabliert ein lokaler Client eine Verbindung mittels COM, ein remote Client nutzt DCOM. OPC Clients und Server können sich auf verschiedenen PCs befinden, sogar wenn sie durch eine Firewall getrennt sind. Um dies zu realisieren, muss DCOM korrekt konfiguriert sein und Ports in der Firewall müssen geöffnet werden. Wird die Windows Firewall benutzt, ist lediglich das Öffnen eines Ports notwendig.

Auch der XML Webservice baut grundlegend auf die Technologien wie auch COM/DCOM jedoch nutzt es als Kommunikationsweg XML via Webservice, wodurch es möglich wurde, OPC auch auf anderen Betriebssystemen zu verwenden. Ein weiterer Vorteil besteht darin, dass die Verbindung wie bei Webservices üblich über den Port 80 kommunizieren und es somit erleichtert wird durch die Firewalls zu kommunizieren.

OPC UA baut auf das Architekturmuster der Serviceorientierten Architekture (engl.: service-oriented architecture) und somit auf die WSDL (Web Service Description Language). Damit wird vor allem die Plattformunabhängigkeit gewärleistet.

[7] Vgl.: [Asco11]

3.1.4 Der OPC-Server

Der OPC-Server ist eine Anwendung, die als API oder Protokollkonverter zu verstehen ist. Der OPC-Server stellt dabei, den OPC-Cients auf Anforderungen Daten und Dienste zur Verfügung. Damit der Server diese bereit stellen kann, benötigt er eine Verbindung zu der Automatisierungsanlage. Der Server und die Steuerung nutzen zur Kommunikation sogenannte „Namespaces “. Ein Namespace ist ein Speicherbereich der Automatisierungsanlage, der verschiedene SPS-Variablen enthält. Eine Variable der Steuerung wird auf dem OPC-Server als sogenanntes OPC-Item repräsentiert.[8]

Ein OPC-Server kann von mehreren OPC-Clients angesprochen werden. Damit steht eine Datenquelle beliebigen OPC-konformen Anwendungen zur Verfügung.

3.1.5 Der OPC-Client

Der OPC-Client nutzt die vom OPC-Server zur Verfügung gestellten Daten und Dienste. Ein Client ist dabei nicht nur auf einen Server beschränkt, sondern kann je nach Leistungsfähigkeit des Systems beliebig viele OPC-Server nutzen. Dabei kann er local oder auf einem entfernten Rechner installiert sein. Local bedeutet, der Server und der Client nutzen den gleichen Rechner, während bei entfernten Rechnern beide auf unterschiedlichen Systemen laufen.

Um die vom Server zur Verfügung gestellten Dienste nutzen zu können, muss der Client eines der beiden zur Verfügung gestellten Interfaces implementieren (siehe Abbildung 3.1). Im Custom-Interface werden die Schnittstellen über die, aus der Programmiersprache C bekannten, Funktionszeiger angesprochen. Weil nicht alle Programmiersprachen mit Funktionszeigern arbeiten können, wird ein sogenanntes „Automation-Interface “ bereitgestellt, mit dem die Objekte über Namen angesprochen werden können. Automation bekommt seinen Namen von dem durch Microsoft geprägten Begriff OLE-Automation, der auf der Grundlage von COM basiert. Damit diese Schnittstelle angewendet werden kann, muss eine sogenannter „Automation-Wrapper “(siehe Abbildung 3.1) eingebunden werden. Dieser Wrapper bildet die Automation-Interface-Aufrufe auf das Custom Interface ab. Der Wrapper, wie auch die Interfaces werden von den OPC-Server-Herstellern in Form von DLL's mitgeliefert. [9]

3.1.6 Das OPC-Item

Ein „OPC-Item“ repräsentiert eine Verbindung zu einer Prozessvariablen der Automatisierungsanlage. Eine Prozessvariable ist ein Element des verwendeten Namespaces

[8] Vgl.: [MaBe11] Seite 17
[9] Vgl.: [MaBe11] Seite 18

des OPC-Servers. Als Beispiel lässt sich der Wert eines Temperatursensors, den der OPC-Server bereitstellt, anführen. Ein OPC-Item wird eindeutig durch seine ID identifiziert . Jedes Item enthält die folgenden Eigenschaften:

- value
- quality of the value
- timestamp

Die Qualität sagt hierbei aus, ob der Wert der Variablen als sicher anzusehen ist. Der timestamp gibt an, wann der Wert des Items das zuletzt ermittelt wurde.

3.1.7 Die OPC-Group

In der „OPC-Group“ werden die vom OPC-Server bereitgestellten OPC-Items strukturiert. Mit Hilfe der Gruppe lassen sich sinnvolle Einheiten von Items bilden und mit diesen Operationen ausführen. So können z.B. alle Prozessvariablen einer Fertigungsmaschine in einer Gruppe zusammengefasst werden.

3.2 Datenbanken

3.2.1 Der Begriff Datenbanken

Die Definition für Datenbanken wird im Taschenbuch der Informatik wie folgt beschrieben:
„Eine Datenbank (engl.: database) ist ein integrierter, persistenter Datenbestand einschließlich aller relevanten Informationen über die dargestellte Information (Metainformation d.h. die Integritätsbedingungen und Regeln), der einer Gruppe von Benutzern in nur einem Exemplar zur Verfügung steht.“[10]

Dabei werden die folgenden Grundforderungen an Datenbanken gestellt:

- Um inkonsistente Daten zu vermeiden, soll eine *Redundante Informationshaltung* vermieden werden. Als Ausnahme lässt sich hier die kontrollierbare Redundanz nennen (Performancegründe).
- Die zweite Grundforderung sieht die Zusicherung von Integritätsbedingungen und Einhaltung von Regeln vor.[11]

Die Datenbanken selbst stellt dabei nur den logisch zusammengehörigen Datenbestand dar. Alle weiteren an Datenbanken gestellte Anforderungen werden durch ein sogenanntes Datenbankmanagementsytem (kurz DBMS) realisiert. Dieses System verwaltet das Datenmodell, die Speicher- und Ladevorgänge, den Mehrbenutzerbetrieb sowie die Sicherheit und den Schutz der Daten. Fasst man das Datenbankmanagementsystem und die Datenbank zusammen so erhält man das Datenbanksystem, das im allgemeinen Sprachgebrauch durch den kürzeren aber wissenschaftlich nicht korrekten Begriff Datenbank geprägt ist.

3.2.2 Relationale Datenbanken

Die Relationalen Datenbanken sind auf den Mathematiker Edgar Frank Codd, einem Mitarbeiter des IBM-Forschungsinstituts in San Jose, CA, zurückzuführen. Er definierte 1970 in „A Relational Model for Large Shared Data Banks[12]“ die grundsätzlichen Prinzipien für das Datenbankmanagement. Nach dieser Definition besteht ein relationales Modell, das sich ausschließlich auf Tabellen stützt, aus drei Komponenten:

- einer strukturellen Komponente die die Datenbank einschließlich der Relationen beschreibt,

[10] [TaBu04] Seite 418
[11] Vgl.: [TaBu04] Seite 418
[12] [Codd70]

- einer manipulierenden Komponente, die aus Operation zur Erstellung von Tabellen besteht,
- und einem Regelsatz, zur Pflege der Integrität in den Datenbanken.

Eine Relation in der Mathematik ist eine Zuordnung von zwei veränderlichen Größen. Dabei ist die eine veränderliche Größe von der anderen veränderlichen Größe abhängig.[13] Somit werden sogenannte Abhängigkeiten zwischen Einträgen erzeugt. Dazu sind zwei weitere Begiffe einzuführen, der Primärschlüssel (engl.: primary key) und der Fremdschlüssel (engl.: foreign key). Der Primärschlüssel darf dabei in jeder Tabelle nur einmal verwendet werden und dient der eindeutigen Identifikation eines Datensatzes. Der Schlüssel kann dabei aus mehreren Attributen zusammengesetzt werden. In der Regel verwendet man jedoch künstliche Schlüssel, die unabhängig von den Attributen sind. Der Fremdschlüssel verweist auf den Primärschlüssel einer anderen Tabelle und muss somit dieselbe Anzahl von Attributen und dem selben Datentypen entsprechen wie der Primärschlüssel.[14]

Relationale Datenbanken können beliebig viele Tabellen enthalten, die in verschiedenster Art und Weise miteinander verknüpft werden können. Informationen werden in Spalten und Reihen abgebildet, wobei eine Reihe einem Datensatz entspricht und eine Spalte einem Attribut des Datensatzes.

Um Mehrdeutigkeiten zu vermeiden, dürfen Spalten sowie Reihen nur einmal vorkommen, es gilt auch zu beachten, dass alle Daten einer Spalte den gleichen Datentyp haben. Da relationale Datenbanken einige Voraussetzungen in Bezug auf die Relation der Daten und deren Extraktion erfüllen, gelten sie als besonders leistungsfähig.

3.2.3 Objektrelationale Datenbanken

Die objektrelationalen Datenbanken halten sich grundlegend an das relationale Datenbankkonzept, dass um bestimmte objektorientierte Konzepte erweitert wurden. Zu diesen gehören die Objektidentitäten und Referenzen, das Typkonzept(„Objekttyp“) erweitert das Tupelkonzept, Kollektionstypen (geschachtelte Relationen), die Vererbungshierarchie. Damit ist es möglich benutzerdefinierte Datentypen zu konstruieren,das bedeutet der Datentyp kann vom Nutzer angelegt werden und muss nicht vom System vorgegeben sein. Hierzu ein kleines Beispiel:

Listing 3.1: Benutzerdefinierte Datentypen

```
1 CREATE TYPE Adress as (
2     street varchar(40),
3     city varchar(40),
4     postalcode char(5),
5     country varchar(40));
```

[13] [BRAI10]
[14] Vgl.: [CJDa03]

```
6
7  CREATE TYPE Person as(
8      firstname varchar(20),
9      lastname varchar(40)
10     birthdate datetime,
11     adress Adress
12 );
```

Wie in dem Beispiel zu sehen ist, wird zunächst der Datentyp Adresse angelegt, der dann in dem Datentyp Person seine Anwendung findet. In den objektrelationalen Datenbanken ist auch eine Vererbung möglich. So kann im folgenden Beispiel der Typ Student von dem zuvor angelegtem Typ Person erben und somit dessen Merkmale erben. Erweitert wird der Student um die Matrikelnummer, die einen Studenten eindeutig macht.

Listing 3.2: Vererbung

```
1 CREATE TYPE Student UNDER Person as(
2     matrikelnumber varchar(10))
3 method GradePointAverage() returns numeric(1,2);
```

Abschließend erwähnt sei noch die typspezifische Methode, die, in diesem Beispiel, unter Verwendung einer bestimmten Rechenvorschrift den aktuellen Notenschnitt des Studenten berechnet. Die so erstellten Typen können als Attributtyp einer Relation oder auch als Tabellentyp verwendet werden. So kann eine Tabelle vom Typ Student mit der folgenden Syntax erstellt werden:

Listing 3.3: Tabelle vom benutzerdefinierten Datentyp

```
1 CREATE TABLE tblStudenten of Student
2     (ref is OID system generated)
```

Dem Datentypen wird dabei ein Objectidentificator, der vom System generiert wurde (system generated) zugewiesen. Die letzte Besonderheit der objektrelationalen Datenbanken ist die Verwendung von Sichten (Views), die von den relationalen Datenbanken abgeleitet wird. Jedoch gibt es hier den Unterschied, dass in der WHERE-Clausel mit der, aus der Programmierung bekannten, Punktnotation auf die Komplexen Objekte zugegriffen wird.

Listing 3.4: Punktnotation im objektrelationalen Modell

```
1 SELECT firstname FROM tblStudenten student
2     WHERE student.lastname='Mustermann';
```

Mit dieser Notation wird auf die Attribute des komplexen Objekttyps zugegriffen werden, dabei spricht man von sogenannten Pfadausdrücken, weil ein Pfad bis hin zum Attribut aufgebaut wird.

Das objektrelationale Modell erweitert also die relationalen Datenbanken um die benutzerdefinierten Datentypen, die Vererbungshierarchien und die Pfadausdrücke.

3.2.4 Objektorientierte Datenbanken

Die relationalen Datenbanken sind das am weitesten verbreitete Datenbanksystem und haben in den letzten 35 Jahren einen beachtlichen Stand erreicht. Der große Vorteil der relationalen Datenbank liegt zum einen in seiner mathematischen Fundierung und zum anderen in seiner Einfachheit. Einfachheit deshalb, weil flache Strukturen problemlos in der Datenbank abgebildet werden können und weil jeder mit gewissen Grundkenntnissen und der Zugriffssprache SQL Abfragen erstellen kann. In dieser Einfachheit liegen aber auch die Grenzen der relationalen Datenbanksysteme.

1987 wurden die ersten objektorientierten Datenbanken veröffentlicht. Objektorientierte Datenbanken basieren auf dem objektorientierten Datenmodell. Dieses Model wurde von Grund auf neu entwickelt und kann jederzeit um neue Datentypen und Funktionen erweitert werden. Die Datendefinition wie auch die Datenmanipulation gehören der objektorientierten Entwicklung an. Die folgenden Funktionen werden durch das objektorientierte Datenmodell bereit gestellt:

- es können die von der Programmiersprache bekannten Datentypen wie z.B. string, integer ohne Konvertierung verwendet werden,
- die Konzepte der Objektorientierung, Klassen, Methoden und Attribute finden ihre Anwendung
- Konzepte zur Beschreibung komplexer Objekte, dazu werden die Typkonstruktoren in die Welt der Datenbanken eingeführt. Dabei wird unterschieden zwischen den Typkonstruktoren „set, list und tuple“. Während set und list mehrere Objekte gleichen Datentyps zu einem Set bzw. zu einer Liste zusammenfassen, kann der Typkonstruktor tuple verschiedene Objekte zu Tuplen wie in Relationen zusammenzufassen.

Mit den Typkonstruktoren lassen sich komplexe Objekte erstellen. Hier ein Beispiel:

Listing 3.5: Typkonstruktoren

```
1 CLASS SeminarGruppe TYPE tuple ( student: Student,
2     Spezialisierung:set (Wahlpflichtfach),
3     Komilitonen: set (Student))
4 method gesamtNotenschnitt: integer
```

In diesem Beispiel geht es, wie der Name schon sagt, um die Seminargruppe. Dabei besteht ein Tuple aus einem Studentenobjekt vom Typ Student. Dieser Student hat Spezialisierungen und Mitstudenten. Da es sich hierbei immer um Objekte vom gleichen

Typ handelt, wird das Set verwendet. Schließlich stellt diese Klasse noch eine Methode, welche den Notenschnitt der ganzen Seminargruppe berechnet.

Ein großer Unterschied zwischen dem objektorientierten Modell und dem klassischen relationalen Datenmodell ist die Objektidentität. Bei der Erstellung eines Objekts in der Datenbank erhält dies einen eindeutigen Identifikator, die Objekt-ID. Diese ID ist in diesem System eindeutig und wird während der Lebenszeit des Objekts nicht geändert. Der Nachteil der relationalen Datenbanksystemen besteht in der Identifikation. Sollte sich an dem Schlüsselatribut etwas ändern, so ist der Datensatz ein völlig anderer. Vermeiden lässt sich dies, durch das Einführen künstlicher Schlüssel. Da beim objektorientierten Modell die ID verborgen bleibt, können alle Attribute geändert werden ohne das sich die Identität des Objekts ändert.

Der Standard für die objektorientierter Datenbanken wird von der ODMG (object data management group) gestellt. Ziele dieses Standards:

- Die Grundbegriffe und die Semantik des Datenmodells werden im Objektmodell festgelegt.
- In der Datenbanksprache ODL werden die Sprachkonstrukte der Datendefinition beschrieben.
- Bei der OQL handelt es sich um eine deklarative Abfragesprache.
- Der Kern des Standard für Objektorientierte Datenbanken sind die Sprachanbindungen. Dabei geht es vor allem daraum, eine möglichst enge Verbindung zwischen der Programmiersprache und der Datenbank herzustellen.

Das Objektmodell der ODMG ist eine Mischung aus Datenmodellen von Java, C++, SmallTalk und den bereits existierenden objektorientierten Datenmodellen. Für den Begriff Klasse wird hier der Begriff Objekttyp eingeführt, dieser kann Werte und Objekte beinhalten. Die Objekttypen werden unterschieden in die veränderbaren(mutable) und nicht veränderbaren(immutable) Objekttypen. Die veränderbaren Objekttypen haben eine Objektidentität, die auf dem System eindeutig ist und sich während der Lebenszeit des Objekts nicht ändert, und einen Zustand, der zu jedem Zeitpunkt seiner Lebenszeit aus der aktuellen Belegung seiner Attribute ergibt. Die Struktur und das Verhalten des Objekts werden durch den Objekttyp festgelegt. Die unveränderbaren Objekte können mit den Datentypen der Programmiersprachen verglichen werden. Sie können entweder einfache Datentypen (integer oder string), oder strukturiert (mit Hilfe von Typkonstruktoren aufgebaut) sein.

3.3 OR Mapping

3.3.1 Der Begriff OR Mapping

Sollen die im Programmablauf erzeugten Objekte die Ausführung des Programms überdauern, so ist es notwendig, diese zu persistieren. Dies ist zum Einen durch Serialisierung und zum Anderen durch das Abbilden der Objekte in Datenbanken möglich. Der Begriff OR Mapping beschreibt eine Technik der Softwareentwicklung, die es ermöglicht, im Hauptspeicher befindliche Objekte der objektorientierten Programmiersprachen, in Datensätzen von relationalen Datenbanken abzubilden.

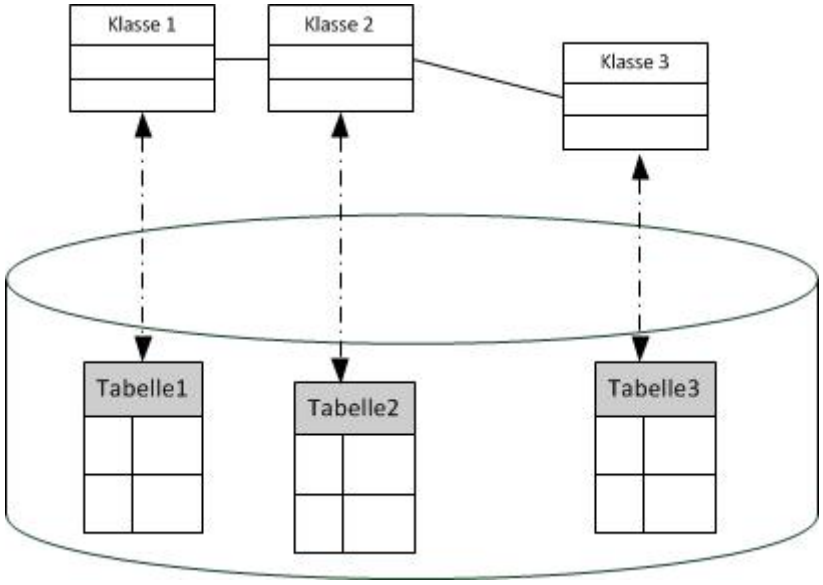

Abbildung 3.2: Das objekt-relationale Mapping

3.3.2 Formen des OR Mappings

Das OR Mapping lässt sich in vier verschiedene Formen gruppieren. Unter diesem Gesichtspunkt werden nachfolgend zwei, unter dem „.NET“, verwendbare OR Mapper analysiert und bewertet.

- *Rein relational* bedeutet, dass die gesamte Anwendung mit dazugehöriger Benutzerschnittstelle, dem relationalen Modell und den auf SQL basierenden relationalen Operationen erstellt wurde. Der Vorteil dieser Form liegt in seiner Performance. Jedoch führt diese Lösung gerade bei großen Projekten zu Unübersichtlichkeit und Schwierigkeiten bei der Wartbarkeit.
- Beim *Light Object Mapping* werden die Klassen von Entitäten repräsentiert. Die Klassen werden manuell in den relationalen Tabellen gemappt abgebildet. Handcodiertes SQL wird vor der Geschäfts-Logik mit Hilfe von Entwurfsmustern verborgen.

- Das *Medium Object Mapping* entspricht einer Applikation, die mit Hilfe eines Objektmodells entworfen wurde. Der SQL-Qelltext wird entweder beim kompilieren mit Hilfe eines Tools zur Codegenerierung oder zur Laufzeit durch das Framework erstellt. Die Beziehungen zwischen den Objekten werden durch eine sogenannte Persistenzschicht unterstützt. Die Abfragen lassen sich durch die verwendete objektorientierte Sprache spezifizieren. Objekte werden durch die Persistenzschicht gecachet.
- Beim *Full Object Mapping* werden Komposition, Vererbung, Polymorphismus und Persistenz durch Erreichbarkeit von der Objektmodellierung unterstützt. Die Persistenzschicht implementiert eine transparente Persistenz. Persistente Klassen müssen weder von speziellen Basisklassen erben noch besondere Interfaces implementieren. Effiziente Fetching- und Caching-Strategien werden in der Anwendung transparent implementiert.[15]

3.3.3 Forward- und Backward Mapping

Das Mappen wird unterschieden in das Forward- und das Backward Mapping. Werden zunächst die Objekte erstellt und aus diesem Objektmodell die Datenbank abgeleitet, so spricht man vom Forward Mapping. Wenn die Objekte aus einer bestehenden Datenbank abgeleitet werden so spricht man von Backwardmapping. Während viele OR Mapper das Konzept des Forward Mappings berherrschen, und immer noch einige das Backward Mapping nutzen, könnnen nur wenige beide Wege.

3.3.4 Das Caching

Da jede Art von Abfrage eine gewisse Abarbeitungszeit benötigt (für komplexere Abfragen können hier schon einmal mehrere Minuten vergehen), wird eine adäquate Cachinstrategie benötigt. Hierfür wird der sogenannte First-Level-Cache eingeführt. Die Daten werden dafür zunächst in diesen Speicher geladen. Bevor es zu einem Datenbankzugriff kommt, wird erst geprüft ob sich die Daten schon im Cache befinden, nur wenn dies nicht der Fall ist, wird aus der Datenbank gelesen. Auch Schreibzugriffe werden zunächst auf dem Speicher ausgeführt und erst abschließend im Datenbanksystem persistiert. Die Vorteile sind dabei:

- eine Steigerung der Performance, weil ein unnötiges mehrfach Laden vermieden wird,
- die Objektrefferenz ist immer die selbe, auch wenn das Objekt mehrfach geladen wird,
- die Daten werden beim mehrfachen Lesen und Schreiben reproduzierbarer.

[15] Vgl.: [KiBa07] Seite

Der Nachteil des Caching ist, dass die Daten zwischen dem Cache und der Datenbank unterschiedlich sein können.

3.3.5 Das Fetching

Das Fetching bezeichnet das Verfahren des Abrufen der Daten aus der Datenbank. Das Fetching findet also zwischen dem Objekt und der Datenbank statt und kann wegen eines oder mehreren Objekten ausgelöst werden. Das Fetching stellt eine Anfrage an das Datenbanksystem, die dann die angeforderten Datensätze sendet. Diese Daten werden mit Hilfe des Mappings übersetzt.

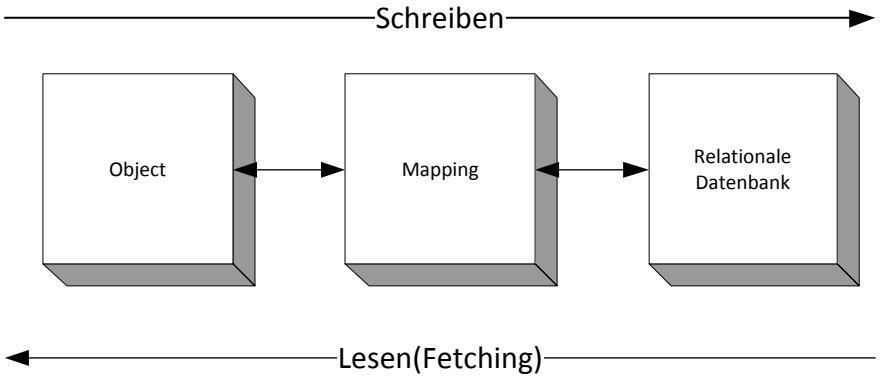

Abbildung 3.3: Das Fetching

3.3.6 Probleme beim OR Mapping

Während Objekte eine eindeutige Identität besitzen und ihre Zustände und das Verhalten kapseln, basiert das relationale Modell auf der aus der Mathematik bekannten relationalen Algebra. Die unterschiedliche Art der Datenspeicherung zwischen Objektmodell und relationalem Modell, wird in der Fachwelt als „object-relational impedance mismatch“ bezeichnet.

Auf dieses Problem aufbauend, wurden verschiedene Lösungskonzepte geschaffen, diese lagen in der Anwendung von objektorientierten Datenbanken, der Erweiterung der Programmiersprachen um die relationalen Konzepte, wie z.B.„Embedded SQL“ und dem hier beschrieben OR Mapping. Das Konzept der objektorierten Datenbanken setzt dieses Problem zwar außer Kraft, scheitert aber zum Einen an seiner schlechteren Performance und zum Anderen daran, dass es bisher kaum verbreitet ist.

Der große Vorteil des Mappens besteht darin, dass beide Konzepte, die objektorientierte Programmierung und die relationale Datenbank, weiterhin ohne Erweiterung bzw. Einschränkung bestehen können. Für die Verknüpfung ist dabei der OR Mapper zuständig, durch ihn werden Regeln festgelegt, wie Objekte in Datenbanken zu persistieren sind.

Da sich die Daten eines Objekts unter normalen Umständen nicht in Tabellen abbilden lassen, müssen diese übersetzt bzw. gemappt werden. Dabei werden im Allgemeinen Klassen auf Tabellen und die aus den Klassen erzeugten Objekte in Zeilen der Tabelle abgebildet.

Es gibt zwei weitere wesentliche Unterschiede zwischen dem Objektmodell und dem Relationalmodell, die Auflösung von „n:m“-Beziehungen und die Vererbung. Das erste Problem ist die Auflösung der Beziehungen. In relationalen Datenbanken bedarf es hierzu einer Zwischentabelle, während diese Beziehung im Objektmodel durch wechselseitige Objektmengen (z.B. typisiert als `List<>` oder untypisiert als `ArrayList`) abgebildet werden kann. Ein guter OR Mapper sollte diese Auflösung beherrschen.

Das zweite Problem ist die **Vererbung**, die das Relationalmodell nicht kennt, kann durch die folgenden Ansätze gelöst werden:

Beim *Vertical Mapping* (auch bekannt als das Joined Mapping) werden sowohl die Basisklasse wie auch alle Unterklassen der Vererbungshierarchie jeweils auf genau eine Tabelle abgebildet. Ein Objekt kann somit nur durch die Verwendung von Joins entstehen. Der Vorteil dieser Methode ist der sparsame Umgang mit dem Speicherplatz, der Nachteil, dass diese Methode, durch die Verwendung der Joins, zeitaufwändiger in der Abfrage ist.

Das *Filtered Mapping* verwendet anders als das Vertical Mapping nur eine Tabelle um die Vererbung abzubilden. In dieser Tabelle werden alle Attribute aller Klassen der Vererbungshierarchie abgebildet, lediglich eine zusätzliche Spalte gibt Aufschluß darüber, welche Zeile zu welcher Klasse gehört. Der Nachteil ist offensichtlich, die Tabelle muss so breit sein, wie die Menge aller Attribute aller Klassen, wodurch viele dieser Zellen immer leer bleiben.

Im *Horizontal Mapping* gibt es Tabellen nur für die konkreten Klassen. Diese Tabellen enthalten alle Attribute der Klassen (auch die geerbten) als Spalte. Um alle Instanzen der Klasse A zu erhalten , ist hierbei nur eine Klasse abzufragen, um jedoch alle Instanzen der Basisklasse A und ihrer Unterklassen B und C zu erfragen muss die Vereinigungsmenge der Tabellen A, B und C gebildet werden.

3.3.7 LINQ To SQL

„LINQ to SQL (Language Integrated Query) ist eine Komponente von .NET Framework Version 3.5, die eine Laufzeitinfrastruktur für die Verwaltung relationaler Daten als Objekte bereitstellt.“[16]
Um zu verstehen was LINQ To SQL tut, muss sich zunächst mit LINQ beschäftigt wer-

[16] [Msdn11]

den.

Was ist LINQ?

In der Programmierung wird mit Daten und Objekten gearbeitet. Interne Daten werden z.B. in Collections gehalten, externe Daten sind zum Beispiel in Datenbanken oder XML-Strukturen abgelegt. Für jede der externen Quellen gibt es eine Abfragesprache (z.B. SQL für Datenbanken). Wird bei der Programmierung auf diese Externen Datenquellen zugegriffen, so muss die Abfragesprache in den Quelltext eingearbeitet werden. Eine Beispielabfrage für den Zugriff auf eine Tabelle einer Datenbank wäre z.B.:

Listing 3.6: Abfrage unter SQL

```
1 SELECT * FROM tblStudents WHERE NAME='MEIER';
```

Davon ausgegangen, dass die Studenten intern in einer Collection gehalten werden, so muss diese Collection mit einer Schleife durchlaufen und jedes Objekt geprüft werden, bis das gesuchte Objekt gefunden wird, dessen Name „Meier“ ist.

Listing 3.7: Abfrage unter C-Sharp

```
1 //Schleife fuer Studenten mit dem Namen Meier
2 foreach(Student student in list)
3 {
4     if(student.NAME==Meier)
5         return student;
6 }
```

LINQ macht es möglich auf „.NET“Objekte mit einer Syntax zuzugreifen, die der von SQL ähnelt. So sieht die gleiche Abfrage unter LINQ wie folgt aus:

Listing 3.8: Abfrage unter LINQ

```
1 //Query fuer Studenten mit dem Namen Meier
2 var student = from s in Students
3                 where s.NAME == "MEIER"
4                 select s;
```

Weitere Vorteile sind: es wird direkt mit Objekten gearbeitet, ein einfaches Debugging ist möglich und die IntelliSense von Visual Studio kann bei der Programmierung genutzt werden. Microsoft bietet derzeit die folgenden Provider an:

- LINQ To Object als Grundlage für alle LINQ Abfragen, damit kann man auf Collections oder Objekte zugreifen,

- LINQ To SQL als Provider für die hausinternen Datenbanksysteme SQL Server 2005 und 2008,
- LINQ To DataSet, dass umfangreiche, optimierte Abfragen der Daten aus einem DataSet ermöglicht,
- LINQ To XML als Schnittstelle zu sich im Arbeitspeicher befindenden XML und
- LINQ To ADO welches aus den beiden separaten Technologien LINQ to DataSet und LINQ TO SQL besteht.[17]

Provider die sich derzeit in der Entwicklung befindenden sind z.B. LINQ To Amazon, LINQ To NHibernate und LINQ to MySQL / Oracle / SQLite.

LINQ To SQL- Abfragen haben eine Besonderheit, sie werden nicht sofort ausgeführt und auch nicht gecachet. Das bedeutet, jede Abfrage wird erst dann und jedes mal wieder ausgeführt, wenn das Ergebnis benötigt wird. Das hat den Vorteil, wenn sich in der Datenbank die Werte geändert haben, so profitiert man von diesem Verhalten. Auf der anderen Seite geht solch ein Verhalten natürlich auf Kosten der Performance. Ob dies als Vorteil oder Nachteil auszulegen ist, ist vom konkreten Einzelfall abhängig.

3.3.8 NHibernate

Nachdem dem Erfolg von Hibernate für Java wurde dieser unter dem Namen NHibernate für das „.NET “ portiert. Wie auch LINQ To SQL gehört NHibernate zu den „Full Object Mapping“ -Formen.

Features von NHibernate

- NHibernate bietet ein *natürliches Programmiermodell* mit Vererbung, Polymorphismus,Komposition und das .NET Collection Framework inklusive der generischen Collections,
- *Native .NET* die API von NHIbernate nutz die .NET Conventionen,
- es wird *kein zusätzlicher Built-time-code* (Quellcode) erzeugt,
- die Software ist *frei erhältlich bzw. Open Source* und steht unter der LGPL (LESSER GNU Public License)
- es wird eine *Zwei-Wege-Kommunikation* geboten, das Programm schreibt nicht nur Daten in die Datenbank, sondern liest sie auch wieder aus dieser,
- NHibernate bietet *Persistenzsicherheit* und übernimmt das Lesen, Schreiben und Locken von Objekten.[18]
- Verfügbar ab dem .NET Framework 1.1.

Als objektorientierte Abfragesprache bedient sich NHibernate einer Eigenentwicklung

[17] Vgl.: [ViCS10] Seite 482
[18] Vgl.: [JBos09]

namens Hibernate Query Language (kurz HQL) wird aber, nachdem es schon einige LINQ-Ansätze verwendet, auch in den nächsten Versionen auf LINQ switchen.[19]

3.3.9 Zusammenfassung

Funktion	LINQ To SQL	NHibernate
Framework	ab .NET 3.5	ab .NET 1.1
Aktueller Release	1.0	3.1.0
Anbieter	Microsoft	JBOSS
Website	www.microsoft.com	www.nhibernate.org
Lizenz	im .NET Framework enthalten	Open Source GNU
Unterstützte Datenbanken	SQL Server 2005, 2008 und MSSQL Compac	MSSQL, DB2, Ingres, Postgres, MySQL,Oracle, Sybase, Firebird,SQLLite
.NET Sprachen	C# und VB	alle
Assistenten	nein	nein
persistiert Felder	ja	nein
persistiert Eigenschaften	nein	ja
verwendete Datenzugriffs-API	ADO.NET	JDBC
Mapping	Forward und Backward	Forward
Form	Full Object Mapping	Full Object Mapping
Mapping bei Vererbung	Filtered Mapping	Filtered-, Vertical-, Horizontal Mapping
Unterstützung von zusammengesetzten Schlüsseln	nein	ja
Abfragesprache	LINQ, SQL	HQL,SQL
Cache	First Level	First und Second Level

Tabelle 3.1: Vergleich der OR Mapper

Wie der Tabelle zu entnehmen ist, sind beide OR Mapper zu den Full Object Mappern zu zählen. Während von NHibernate alle gängigen Datenbanken unterstützt werden, bietet Microsoft zur Zeit nur die Datenbanken aus dem eigenem Haus. Abhilfe wird geschaffen, durch Provider, die noch in der Entwicklung sind, wie z.B. LINQ To Oracle usw. Zu den zwei Wege Mappern lässt sich nur LINQ zählen da von NHibernate nur das Forwardmapping unterstützt wird.

[19] Vgl.: [AyRa07]

4 Das Lösungskonzept

Die Anforderungen an das Framework wurden in dem Kapitel 2.2 benannt. Zur Entwicklung einer Software ist ein sogenannter Softwareentwicklungsprozess notwendig. Nach der Erläuterung des Begriffs, wird ein Vorgehensmodell zur Entwicklung des OPC-OR-Mappers gewählt. Daraufhin wird ein Datenbankmanagementsystem für das Framework gewählt. Anschließend werden die im Kapitel Technologischen Grundlagen vorgestellten OR-Mapper in die Entscheidung zur Wahl eines OR-Mappers eingebunden.

Abschließend werden die Anforderungen, welche beim Entwurf der Klassenbibliothek berücksichtigt werden sollen, beschrieben.

4.1 Softwareentwicklungsprozess

Der Softwareentwicklungsprozess wird vom Fraunhofer-Institut[20] wie folgt beschrieben:

> Die Entwicklung oder Anpassung komplexer (Software-) Systeme erfordert Klarheit über das zu erreichende Ziel, eine gemeinsame Sicht aller beteiligten Parteien auf das System, eine definierte Vorgehensweise zur Lösung der Probleme und einen definierten Output in jeder Phase des Projekts. Sind diese Punkte nicht erfüllt, kann es zu hohen Kosten, Verzögerungen oder gar zum Scheitern des Projekts kommen.
>
> Der Softwareentwicklungsprozess muss planbar, nachvollziehbar und kontrollierbar sein. Um mehr Transparenz zu schaffen, wird ein Vorgehensmodell eingeführt.

4.1.1 Vorgehensmodelle

Das Vorgehensmodell ist eine Sammlung von aufeinander abgestimmten Richtlinien, die die Durchführung von Projekten beschreiben. Es umfasst Tätigkeiten des Software-Entwicklungsprozesses, Artefakte (Dokumente usw.), die während des Entwicklungsprozesses entstehen sowie die Rollen von Personen. Man unterscheidet drei Standards für Vorgehensmodelle:

- sequentiell oder phasenorientiert
- iterativ oder evolutionär
- leichtgewichtig oder agil

[20] [FraHo11]

Der sequentielle bzw. phasenorientierte Ansatz

Im sequentiellen Ansatz wird ein sequentielles Vorgehen beschrieben. Es gibt klar definierte Phasen und jede dieser Phasen liefert ein Ergebnis (z.B. liefert die Analyse das Lastenheft). Bekannte Modelle dieser Gattung sind das Wasserfall-Modell und das V-Modell.

Vorteile:

- das Modell ist einfach durchzuführen,
- wenn die Anforderungen des zu bewältigenden Projektes klar gesteckt sind, ist das Modell sehr effizient,
- es bietet eine gute Möglichkeit der Überwachung.

Nachteile:

- es ist während des Projektes relativ starr,
- Probleme aus zurückliegenden Phasen können sich bis zum Schluss verschleppen.

Das iterative bzw. evolutionäre Vorgehensmodell

Das Spiral-Modell und der Rational Unified Process sind die bekanntesten Vertreter dieser Gattung. Iterativ bedeutet dass, anders als beim Wasserfallmodell, der Prozess mehrmals (iterativ) durchlaufen wird. Es werden sozusagen kleine Wasserfälle hintereinander gesetzt. Dieses Vorgehenmodell ist eine Weiterentwicklung des sequentiellen Ansatzes und folgt der Erkenntnis, dass Software auch vom Funktionsumfang gepflegt werden muss. Das zu erstellende System wird nicht mit einem mal sondern in mehreren Stufen (inkrementell) freigegeben.

Wenn die Entwicklung einem beliebigen Ansatz folgt und kontinuierlich weiterentwickelt wird, so spricht man vom evolutionärem Modell.

Vorteile:

- sind vor allem, dass die Risiken des sequentiellen Modells früher erkannt werden,
- wechselnde Anforderungen können besser und schneller berücksichtigt werden,
- die inkrementelle Auslieferung wird erleichtert.

Nachteile:

- sind vor allem die Mehrarbeit,
- ein komplexeres Projektmanagement,

- und es ist schwerer nachvollziehbar als das sequentielle Modell.

Das adaptive(oder agile) Modell

Zu den bekanntesten Vertretern dieses Modells gehören das eXtreme Programming und SCRUM. Der adaptive Software-Process ist eine Weiterentwicklung des iterativen Ansatzes. Der Unterschied liegt in der dynamischen Planung wodurch eine kontinuierliche Anpassung an Änderungen gewärleistet werden kann.

Vorteile:

- es ist bei unklaren Zielen und sich ändernden Anforderungen gut einsetzbar,
- es verspricht ein besseres Kosten/Nutzen-Verhältnis,
- es wird davon ausgegangen, dass die durchschnittliche Code-Qualität(z.B. durch Pair Programming) besser ist

Nachteile:

- ein Ergebnis ist nicht immer vorhersagbar,
- auch die Qualitätseigenschaften können nicht garantiert werden,
- weiterhin ist häufig nicht nachvollziehbar, wie eine Funktion zustande kommt,
- da ca. 80-90Prozent aller Software auf eingebetteten Systemen (z.B. Maschinen) läuft und Fehler fatal wären, findet das agile Modell hier keine Anwendung.
- Refactoring wird für die objektorientierten Sprachen nicht unterstützt.

4.1.2 Wahl des Vorgehensmodells

Das klassische Wasserfallmodel lässt sich unter den sequentiellen Vorgehensmodellen einordnen. Positiv zu bewerten ist dabei, dass jede Phase etwas liefert und somit immer der derzeitige Stand kontrolliert werden kann. Somit sind gerade in der Phase der Konzeption Entwicklungen in die falsche Richtung ausgeschlossen. Ein Risiko hierbei liegt jedoch im Test, denn erst hier werden Fehler, die während der Implementierung gemacht wurden, ersichtlich. Die Fehlerbeseitigung kann sich dann in einzelnen Fällen als sehr umfangreich gestalten. Das Wasserfallmodell ist also, auch bedingt durch das Alter, nur bei Projektem mit einem geringen Aufwand (Manntagen) und fest vorgegeben Anforderungen zu empfehlen.

Bei großen und komplexen Projekten orientiert sich viele Vorgehensmodelle am sogenannten V-Modell. Durch das zeitige Definieren der Testfälle wird das Ziel verfolgt, das Projektrisiko zu minimierne und damit die Softwarequalität und die Transparenz zu erhöhen. Dabei sollte das Model, durch die Schrittweise Entwicklung (inkrementell), mit mehrfach durchlaufen(Iterationen) werden.

Die agilen Vorgehensweisen, z.B. SCRUM, gehen davon aus, dass das Projektziel aufgrund seiner Komplexität nicht im Voraus bis ins Detail geplant werden kann. Im Ablauf eines Projektes ist es daher besser, wenn ein Team organisiert wird, das die Verantwortung für die Fertigstellung der einzelnen Teilergebnisse übernimmt. Weil gerade die agilen Vorgehensweisen auf einer intensiven Kommunikation zwischen Auftraggeber und Auftragnehmer basieren, wird das zeitliche Vorgehen, im Ablauf, in kurze Zyklen (die sogenannten Sprints) geteilt. Am Ende eines jeden Zyklus ist das Feedback des Auftraggebers notwendig. Aufgrund der Kommunikation sind lange Entscheidungswege als problematisch anzusehen und sprechen somit gegen das agile Modell.[21]

Fazit: **Das richtige Vorgehensmodell gibt es nicht!**

Es ist jedoch möglich, die Vorgehensmodelle anzupassen. Gerade umfangreiche Vorgehensmodelle wie das V-Modell XT müssen vor ihrem Einsatz an das Projekt angepasst werden. Durch das sogenannte Tailoring (deutsch: schneidern) werden die Teile entfernt, die nicht zum Projekt passen. Ziel des Tailoring ist es, dass nur so viele Regeln und Vorgaben definiert werden, wie auch wirklich benötigt werden. Jedoch müssen alle notwendigen Regeln auch tatsächlich eingehalten werden.[22]

Da bei dem zu realisierendem Projekt von einem Einmann-Projekt ausgegangen wird und gleichzeitig der Bezug zum Auftraggeber vorhanden ist, fiel die Wahl auf das V-Modell. Die Vorteile liegen in der Iteration der Entwicklungsphase. Ziel ist es, das Produkt in Paketen zu entwickeln. Diese Pakete werden nach Klassen durch die Verwendung eines Testframeworks getestet. Somit kann die Funktion einer jeden Klasse von vornherein sicher gestellt werden und der „große Knall“ am Ende des Projektes bleibt aus. Dennoch sind auch bei der Wahl dieses Modells Anpassungen notwendig. So ist es von Seiten des Auftraggebers notwendig, die im Wasserfallmodell realisierten Dokumente, in das Modell zu übernehmen um eventuelle Fehlentwicklungen zu vermeiden. Damit wird das V-Modell wie in der Abbildung zu sehen abgeändert.

[21] Vgl.:[MoSc10]
[22] Vgl.:[MoSc10]

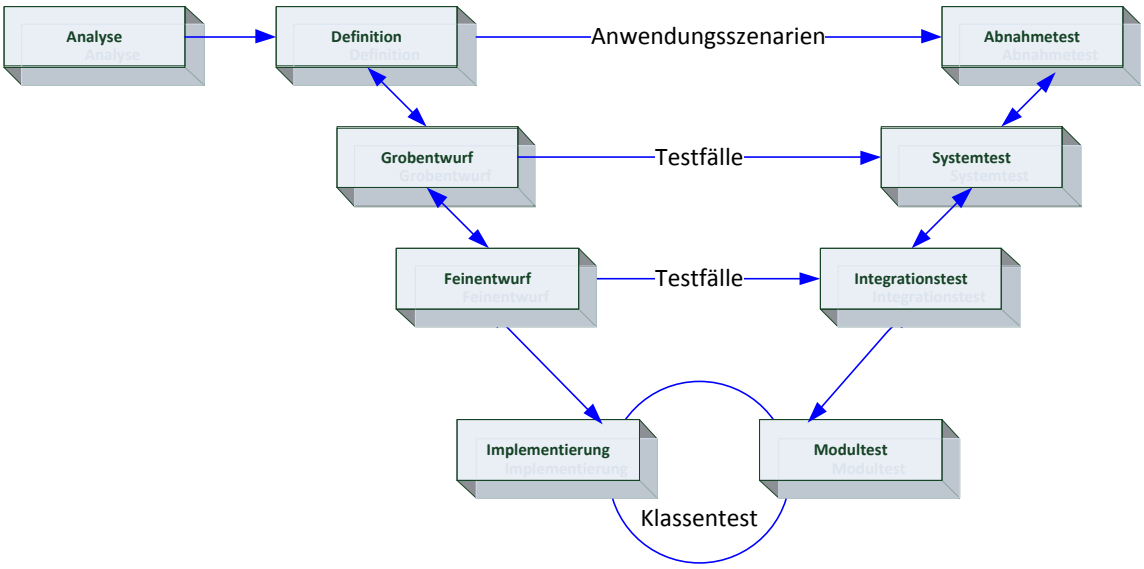

Abbildung 4.1: Das angepasste V-Modell

In der Abbildung ist zwischen der Implementierung und den Modultests eine Iteration zu erkennen, dies ist so zu deuten, dass jede Klasse nach der Fertigstellung ein Pendant als Testklasse erhält erst nachdem alle Methoden der Klasse erfolgreich getestet wurde, wird die nächste Klasse implementiert. Nachdem ein Paket fertiggestellt wurde, wird auch dies einem Test unterzogen. Für die Modultests wird das Testframework NUnit, in seiner aktuellen Version 2.5.10, ausgewählt.

4.2 Auswahl eines Datenbankmodells

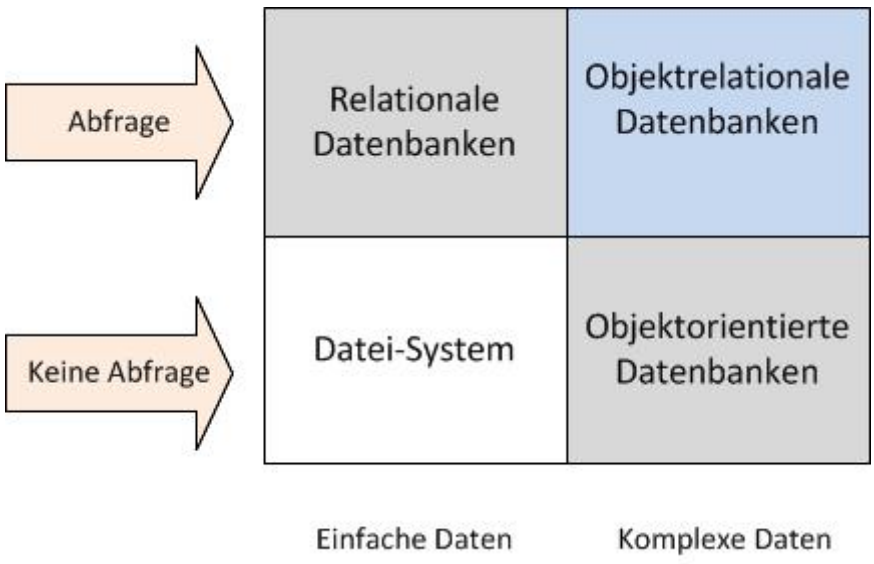

Abbildung 4.2: Matrix zur Wahl des Datenbanksystems

Eine Entscheidung zur Wahl des Datenbankmodells lässt sich mit Hilfe der Matrix in der Abbildung 4.2 herbeiführen. Ausgegangen von komplexen Daten und davon, dass keine Abfragen an das Datenbanksystem gestellt werden fällt die Wahl auf die Objektorientierten Datenbanken (kurz: OODB). Ein großer Vorteil hierbei ist, dass sich so die Typkonvertierungen sparen lassen. Dennoch lässt sich dieser Vorteil nicht gegen die Performance eines relationalen Datenbanksystems aufwiegen. Auch die Verbreitung der objektorientierten Datenbanken stellt ein großes Problem bei deren Anwendung dar. Da die HGDS-GmbH sehr kundenorientiert arbeitet und bei ihren Kunden nur relationale Datenbanksysteme zum Einsatz kommen, fällt die Entscheidung des Datenbanksystems trotz des benannten Vorteils auf die relationalen Datenbanken.

4.3 Auswahl eines OR Mappers

Die Tabelle 3.1 gibt einen Aufschluss über die untersuchten OR Mapper. Bei der Suche nach einem geeigneten Produkt sind die folgenden Punkte zu beachten. Der OR Mapper sollte:

1. SQL Server und Oracle Datenbanken unterstützen,
2. das Framework 2.0 unterstützen,
3. zu der Gruppe der Light Object Mapping Formen gehören und
4. das Forwardmapping beherrschen.

In der aktuellen Version unterstützt nur der NHibernate Mapper die beiden Datenbanksystem SQL Server und Oracle. Das Framework 2.0 wird nur von NHibernate unterstützt. Beide betrachtenen OR-Mapper gehören zu der Gruppe der Full Object Mapper, was auch das gefordertre Light Object Mapping einschließt. Auch das Forward-Mapping wird von beiden betrachteten Lösungen unterstützt.

Dennoch geht die Entscheidung dahin, dass der OR-Mapper eine Eigenentwicklung sein sollte, dafür sind die folgenden Gründe zu nennen:

- Aus kaufmännischer Sicht gilt es die Lizenz zu beachten, denn anders als bei dem Pendant für Java (Hibernate) steht NHibernate unter der GPL (General Public License). Die Regeln dieser Lizens besagen, dass alle abgeleiteten Produkte eines unter der GPL stehenden Werkes nur dann vertrieben werden, wenn dies ebenfalls unter der GPL-Lizensierung geschieht. Da das zu erstellende Produkt jedoch auch für den Vertrieb bereitstehen soll, kann diese Bedingung nicht akzeptiert werden.
- Aus technischer Sicht können nur OR Mapper verwendet werden, die für das Framework 2.0 verfügbar sind. Um die Anwendung schlank zu halten sollten nur Funktionen implementiert werden, die auch tatsächlich benötigt werden. Dies kann nicht gewärleistet werden, wenn ein Fremdprodukt Anwendung findet, da

dies Funktionen mitbringt, die in dem zu erstellenden Produkt keine Anwendung finden werden.

4.4 Verwendete OPC-Spezfikation

Im Kapitel Technologische Grundlagen wurden die verschiedenen OPC-Spezifikationen vorgestellt. Die Standards die in der Entwicklung des OPC-OR-Mappers Anwendung finden können sind:

- OPC DA als Standard für die Datenübertragung, der auf Microsofts Technologie COM bzw. DCOM baut,
- OPC XML DA als Spezifikation zur XML-basierten Übertragung (aufbauend auf OPC DA) und
- OPC UA als Ablösung des OPC DA Standards.

Für die Entwicklung wird hierbei auf den Standard OPC DA zurückgegriffen. Da OPC XML DA aufgrund der Entwicklung des neuen Standards OPC UA kaum Verbreitung fand, können nur die anderen beiden Spezifikationen in die Entscheidung einfließen. Die Gründe gegen den neuen OPC UA Standard sind allen voran, daß noch nicht alle Hersteller von OPC-Server auf den neuen Standard bauen. Viele Hersteller schwenken erst nach und nach auf diese neue Technik um. Dabei erfüllt die OPC DA Spezifikationen immer noch alle Anforderungen der derzeitigen Standards und kann somit bedenkenlos verwendet werden.

4.5 Verwendung einer Datenzugriffsschicht auf OPC-Server

Da ein Grundgedanke der objektorientierten Programmierung die Wiederverwendbarkeit ist und die HGDS-GmbH bereits eine Klassenbibliothek entwickelt hat, die den Datenaustausch mit OPC-Servern, unter Verwendung der OPC-Spezifikation OPC DA, möglich macht, fällt die Wahl der Datenzugriffsschicht auf dieses Paket. Zum Zugriff auf den OPC-Server stehen dabei zwei Möglichkeiten zur Verfügung:

- Die Klasse `OpcClient`, die den im Kapitel 3.1.5 erläuterten Wrapper verwendet. Diese Klasse stellt zwar alle notwendigen Methoden zur Verfügung, bedarf aber eines größeren Konfigurationsaufwands.
- Die Klasse `OpcClientWizard`, die auch Verwendung in der Entwicklung finden wird, nutzt die Klasse OPC-Client um auf den OPC-Server zuzugreifen und stellt Möglichkeiten zum Herstellen und Trennen einer Verbindung zu OPC-Servern bereit. Weiterhin lassen sich mit dieser Klasse Items lesen und schreiben. Damit

sind alle Anforderungen an den Zugriff auf den OPC-Server erfüllt und müssen nur noch angepasst werden.

4.6 Verwendung einer Datenzugriffschicht auf die Datenbanken

Unter der Datenzugriffschicht versteht man eine effektive Schnittstelle zwischen Anwendungslogik und Datenbanksystemen.[23] Dazu wird die, an der Hochschule-Mittweida von Christian Huschto[24] als Diplomarbeit eingereichte, Datenzugriffsschicht verwendet. Diese wurde für die HGDS-GmbH entwickelt und erfüllt die folgenden Anforderungen:

- Unterstützung mehrerer Datenbanksysteme.
- Zeitnahe und zuverlässige Ausführung von Datenbankoperationen.
- Sparsame Verwendung und zuverlässige Freigabe von Resourcen.
- Asynchrone Ausführung von Operationen.
- Überwachung der Verbindung zur Datenquelle.
- Wiederholung von fehlgeschlagenen Operationen.[25]

Eine Anweisung unter Verwendung der vom Framework bereit gestellten Methoden würde exemplarisch wie folgt aussehen:

Listing 4.1: ExecuteNonQuery unter dem Framework

```
1  private static void CreateCommand(string queryString,
2      string connectionString)
3  {
4      using (SqlConnection connection = new SqlConnection(
5                  connectionString))
6      {
7          SqlCommand command = new SqlCommand(queryString, connection);
8          command.Connection.Open();
9          command.ExecuteNonQuery();
10     }
11 }
```

Dabei muss zunächst eine Verbindung erstellt werden, bevor die eigentliche Befehlsausführung ablaufen kann. Durch die Verwendung der Datenzugriffsschicht wir die Verbindung einmal festgelegt (kann aber auch jederzeit getauscht werden). Somit sieht die gleiche Anweisung wie folgt aus:

Listing 4.2: ExecuteNonQuery unter der Datenzugriffsschicht

```
1  this._dataAccess.ExecuteNonQuery(CommandType.Text, queryString,
```

[23] Vgl.:[CHHG06] Kapitel 1.1
[24] [CHHG06]
[25] [CHHG06]

```
2                                   iDataParameter.Parameters);
```

Die Datenzugriffsschicht erleichtert dem Entwickler den Umgang mit den Datenbanken.

4.7 Zusammenfassung

Das Produkt wird unter Verwendung der OPC Data Access Spezifikation und dem relationalen Datenbankmodell entwickelt. Eine Übersicht des zu erstellenden Produktes gibt das Umweltdiagramm in der Abbildung 4.3. Schnittstellen existieren zum Einen zwischen dem Framework und den Datenbanken und zum Anderen zwischen dem Framework und den OPC-Servern. Die beiden Schnittstellen nutzten zur Kommunikation die in den vorherigen Abschnitten beschriebenen Datenzugriffsschichten zum OPC-Server und die von Christian Huschto[26] erstellte Schicht für die Datenbanken.

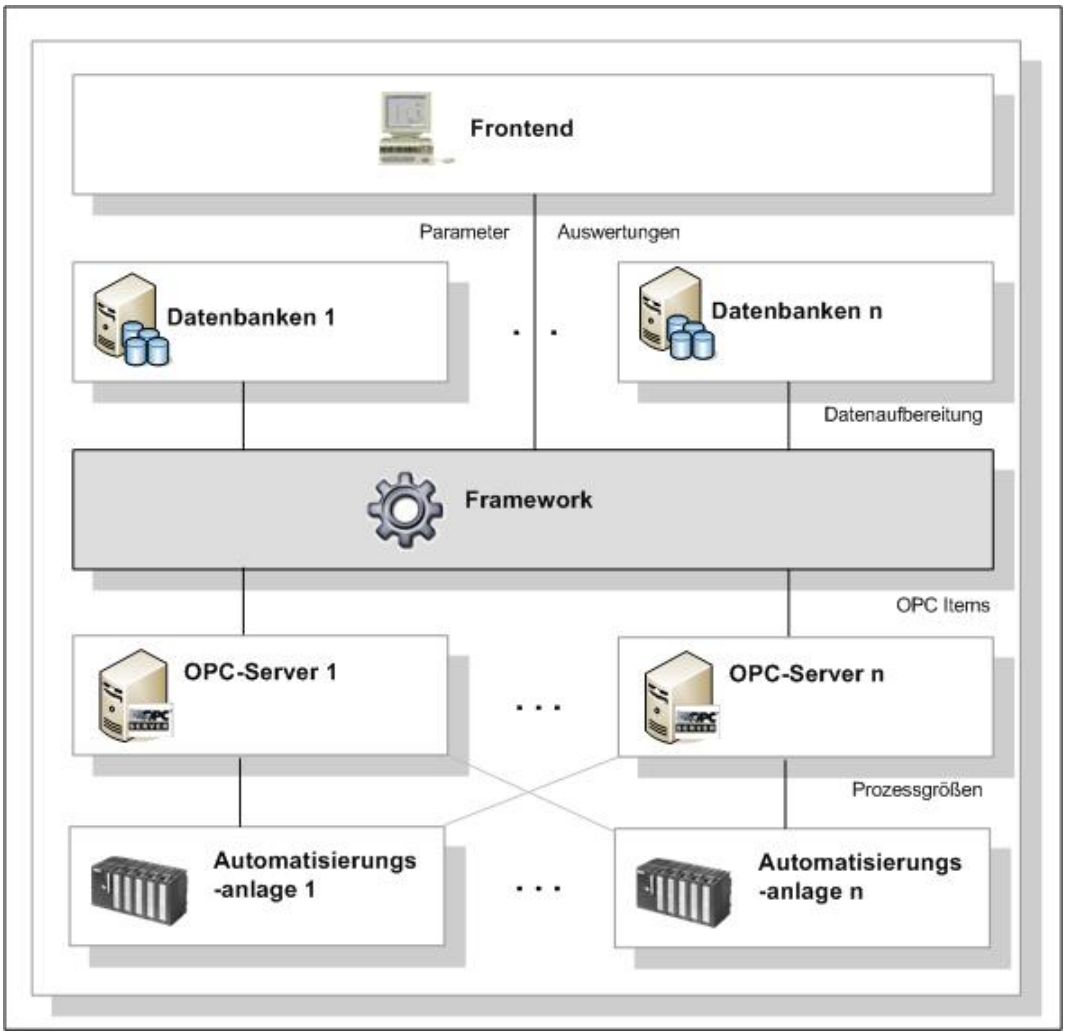

Abbildung 4.3: Umweltdiagramm

[26] [CHHG06]

5 Die Umsetzung des Prototypen OPC-OR-Mapper

Nachdem die Ziele und Anforderungen in den zurückliegenden Kapiteln benannt wurden, soll das Framework und das dazugehörige Frontend entwickelt werden. Dazu wird das im Lösungskonzept vorgestellte Vorgehensmodell angewendet.

5.1 Die Anwendung des Vorgehensmodells während der Entwicklung

Der Entwicklungsprozess folgt dem vorgestelltem Vorgehensmodell mit den Phasen Analyse, Definition, Entwurf, Implementierung und den zugeordneten Testverfahren. Die Phasen werden während des Entwiklungsprozesses durchlaufen. Jede Phase liefert dabei ein Resultat, dass dem Auftraggeber vorgelegt werden kann, um einer möglichen Fehlentwicklung entgegenzuwirken bzw. diese frühzeitig zu erkennen.

Eine Besonderheit bei dem angewendeten Vorgehensmodell ist die Zerlegung in sogenannte Teilsysteme. Diese Zerlegung ist der Modularisierung geschuldet und bietet den Vorteil, dass so Komponenten jederzeit ersetzt werden können.

Wenn mit der Zerlegung dieser Teilsysteme begonnen wird, bedeutet dies, dass auch auf der Integrationsseite des V- Modells diese Ebenen hinzugefügt werden müssen und im jeweiligen System Tests durchgeführt werden müssen.

Das angepasste Vorgehensmodell beachtet dies, der Entwicklungsprozess wird daher iterativ durchlaufen. Dies bedeutet, daß jedes System mit seinen Klassen einzeln implementiert wird. Nachdem eine Klasse fertiggestellt wurde, wird diese, unter Verwendung eines Testframeworks, auf Funktion getestet. Sind die Implementierung eines Teilsystems abgeschlossen und alle Klassen erfolgreich getestet, so wird das System dem Modultest unterzogen.

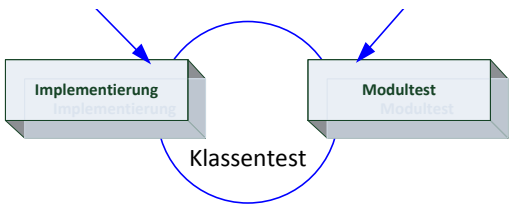

Abbildung 5.1: Das angepasste V-Modell

Abgeschlossen wird die Entwicklung durch den Integrationstest, den Systemtest und

den Abnahmetest.

5.2 Die Beschreibung der Anforderungen in der Analyse durch das Lastenheft

Das Lastenheft enthält die Anforderungen des Auftraggebers an die Lieferung bzw. Leistung des Auftragnehmers. Darin enthalten sind die Anforderungen aus der Sicht des Anwenders. Es definiert „Was“ für eine Aufgabe erstellt werden soll und „Wofür“ diese zu lösen ist.[27] Dabei gilt es zu beachten, dass die Anforderungen auf die fundamentalen Funktionen zu beschränken und präzise, ohne zu weit ins Detail zu gehen, zu formulieren sind.[28] Das Lastenheft hält sich dabei an die von Helmut Balzert[29] vorgeschlagene Gliederung.

5.2.1 Zielbestimmungen

Die Firma HGDS GmbH ist in den beiden Sparten Automatisierung und Softwareentwicklung tätig. In diesem Zusammenhang ist es für die Softwareabteilung der Firma notwendig, Anwendungen zu entwickeln, die in der Lage sind, mit Steuerungen zu kommunizieren. Dabei müssen in diesen Werte ausgelesen und geändert sowie weiteren Prozessen zur Verfügung gestellt werden. Bisher wurden für die realisierten Projekte individuelle Lösungen erstellt. Ziel dieses Projektes ist es, einen einheitlichen Lösungsansatz zu schaffen. Dabei gilt es die beiden grundlegenden Funktionen Konfiguration und den Datentransfer zu beachten.

Das zu erstellende Paket soll die Entwicklung von Automatisierungslösungen bei der Firma HGDS in den folgenden Punkten unterstützen:

- Konfiguration der Schnittstelle zur Steuerung
- Konfiguration der Schnittstelle zur Datenbank
- Auswahl der Daten, die aus der Steuerung ausgelesen werden sollen
- Verwaltung des Datentransfers
- Erstellung eines Konfigurationsscripts für den OPC-Server

5.2.2 Produkteinsatz

In erster Linie wird dieses zu erstellende Produkt zur Projektierung und Inbetriebnahme durch autorisiertes Personal verwendet. Dadurch kann in dieser Ausbaustufe auf eine

[27] Vgl.:[ITHa06] Seite 335
[28] Vgl.:[HeBa00] Seite 63
[29] [HeBa00] Seite 62

Benutzerverwaltung verzichtet werden.

Die unter den Zielbestimmungen benannten Aufgaben sollen durch ein GUI ergänzt werden und ohne jeglichen Programmieraufwand getätigt werden können. Zusätzlich soll die Programmoberfläche einfach gehalten werden und gleichzeitig intuitiv bedienbar sein.

Das Paket ist modular zu gestalten, um die Anforderungen bezüglich der Erweiterbarkeit und Änderbarkeit zu erfüllen.

Da das Programm in Deutschland verwendet wird, dient die deutsche Sprache als Verkehrssprache.

5.2.3 Produktübersicht

Die Produktübersicht enthält die Funktionen des OPC-OR-Mappers.

Abbildung 5.2: UseCase Produktübersicht

5.2.4 Produktfunktionen

Übersicht

Pos	Name	Kennung	Beschreibung
1	/LF010/	Funktion: Akteur: Beschreibung:	Verbindung zur Datenbank herstellen Administrator Anhand der Eingaben des Akteurs soll eine Verbindung zur Datenbank hergestellt werden. Hierzu werden die bereits vorhandenen Datenbankschnittstellenklassen verwendet.
2	/LF020/	Funktion: Akteur: Beschreibung:	Verbindung zur Datenbank ändern Administrator Eine bestehende Verbindung soll geändert werden.
3	/LF030/	Funktion: Akteur: Beschreibung:	Verbindung zur Datenbank löschen Administrator Eine Verbindung zur Datenbank soll sich erst nach Rückfrage löschen lassen, da hierdurch der Verkehr zwischen OPC-Server und Datenbank beeinträchtigt wird.
4	/LF040/	Funktion: Akteur: Beschreibung:	Verbindung zum OPC-Server herstellen Administrator Aufbau einer Verbindung zu einem OPC-Server anhand der zu erfassenden Parameter.
5	/LF050/	Funktion: Akteur: Beschreibung:	Verbindung zum OPC-Server ändern Administrator Der Administrator ändert eine bestehende Verbindung zum OPC-Server.
6	/LF060/	Funktion: Akteur: Beschreibung:	Verbindung zum OPC-Server löschen Administrator Da sich das Löschen der Serververbindung auf die Kommunikation zwischen dem OPC-OR-Mapper und der Datenbank auswirkt, ist das Löschen zuvor zu bestätigen.
7	/LF070/	Funktion: Akteur: Beschreibung:	Verwaltung der zu mappenden Daten Administrator Die vom OPC-Server vorliegenden Daten sollen mit dieser Funktion aus- bzw. abgewählt werden.
8	/LF080/	Funktion: Akteur:	Erstellung einer Konfiguration Administrator

Pos	Name	Kennung	Beschreibung
		Beschreibung:	Auf der Grundlage der zuvor erstellten Verbindungen, zur Datenbank und zum OPC-Server und der selektierten zu mappenden Daten soll es mit dieser Funktion möglich sein, eine Konfiguration zu erstellen.
9	/LF090/	Funktion: Akteur: Beschreibung:	Ändern einer Konfiguration Administrator Diese Funktion schließt die Funktionen Datenbankverbindung ändern, OPC-Serververbindung ändern sowie die Verwaltung zu mappenden Daten ein.
10	/LF100/	Funktion: Akteur: Beschreibung:	Löschen einer Konfiguration Administrator Eine Konfiguration soll nach Bestätigung des Löschvorgangs entfernt werden.
11	/LF110/	Funktion: Akteur: Beschreibung:	Datenaustausch Administrator Nachdem die Konfiguration erstellt wurde, soll unter Auswahl einer geeigneten Methode der Datentransfer erstellt werden. Hierfür stehen die unter dem Punkt A.4.3 (Datenaustausch) benannten Methoden zur Verfügung.
12	/LF120/	Funktion: Akteur: Beschreibung:	Daten aus der Steuerung lesen System Angestoßen durch das Starten des Datentransfers sollen die konfigurierten Datenquellen mit Hilfe des OPC-Servers direkt aus der Steuerung gelesen werden. Die Aktualisierung der Daten ist dabei abhängig von der im Datenaustausch gewählten Methode.
13	/LF130/	Funktion: Akteur: Beschreibung:	Daten verarbeiten System In dieser Funktion finden die konfigurierten Anpassungen der ausgelesenen Werte (z.B. Typkonvertierungen) an die Struktur der Datenbank statt.
14	/LF140/	Funktion: Akteur: Beschreibung:	Daten in der Datenbank speichern System Nach der Verarbeitung der Daten sollen diese, unter Berücksichtigung der in der Konfiguration festgelegten Verknüpfungen, in der Datenbank persistent abgelegt werden.

Tabelle 5.1: Produktübersicht

Konfiguration

In der Konfiguration werden zunächst über Dialoge die Quelle (OPC-Server) , und das Ziel (die Datenbank) konfiguriert. Nach erfolgreicher Verbindung zum OPC-Server werden alle verfügbaren Werte ausgelesen und der Konfiguration zur Verfügung gestellt. Abschließend lassen sich die so selektierten Daten der Quelle mit den Daten des Ziels verbinden.

Datenaustausch

Der Datenaustausch ist das Resultat der zuvor erstellten Konfiguration. Dabei sollen die Daten abhängig von der gewählten Aktualisierungsmethode aus dem OPC-Server gelesen und in die Datenbank geschrieben werden. Zur Aktualisierung stehen die folgenden Methoden zur Verfügung:

- Triggerbasierte Aktualisierung
  Nachdem die SPS die Daten vollständig geschrieben hat, wird innerhalb der Steuerung ein Bit auf den Wert „Eins“ gesetzt. Der Vorteil dieser Methode ist, dass lediglich dieses Bit und nicht einzelne Werte überwacht werden müssen.
- Intervallbasierte Aktualisierung
  Der Datenaustausch findet hierbei zyklisch in einem zuvor festgelegten Intervall statt.
- Änderungsbasierte Aktualisierung
  Jeder Wert innerhalb des Datenaustausch wird überwacht. Ändert sich einer dieser Werte, so gilt dies als Auslöser für das Schreiben des Wertes in die Datenbank
- Grenzwertbasierte Aktualisierung
  Bei der grenzwertbasierten Aktualisierung ist es notwendig, Werte für eine obere bzw. untere Schranke festzulegen. Das Übernehmen der Daten aus der Steuerung und das Schreiben der Werte in die Datenbank wird dabei ausgelöst, sobald diese Werte über- bzw. unterschritten werden.

5.2.5 Produktdaten

Pos	Name	Kennung	Beschreibung
1	/LD010/	Name der Datengruppe: Maximale Anzahl: Beschreibung:	Datenverbindung 100 Informationen zur Datenverbindung • Bezeichnung • Host • User • Passwort • Name der Datenbank • Typ der Datenbank
2	/LD020/	Name der Datengruppe: Maximale Anzahl: Beschreibung:	OPC-Server-Verbindungen 100 Informationen zur OPC-Server-Verbindung • Bezeichnung • Host • Port • Typ des Servers
3	/LD030/	Name der Datengruppe: Maximale Anzahl: Beschreibung:	Konfiguration 15 Informationen zur Konfiguration • Bezeichnung • Datenbankverbindung • OPC-Server-Verbindung • Datum • Typ der Überwachung
4	/LD040/	Name der Datengruppe: Maximale Anzahl: Beschreibung:	Item 1.000.000 Informationen zum erfassten Item

Pos	Name	Kennung	Beschreibung
			• Bezeichnung • Konfiguration • OPC-Name • Datentyp • Wert • DB-Name • Beschreibung • TimeStamp

Tabelle 5.2: Produktdaten

5.2.6 Produktleistungen

Pos	Name	Kennung	Beschreibung
1	/LL010/	Beschreibung:	Transaktionen nach dem ACID

Tabelle 5.3: Produktleistungen

5.2.7 Qualitätsanforderungen

Pos	Name	Sehr gut	Gut	Normal	Nicht relevant
1	Funktionalität	X			
2	Zuverlässigkeität	X			
3	Benutzbarkeität			X	
4	Effizienzät			X	
5	Änderbarkeität	X			
6	Übertragbarkeität	X			

Tabelle 5.4: Produktleistungen

5.3 Die Umsetzung der Anforderungen in der Definition

Nachdem der Auftraggeber, in der Phase Analyse, seine Anforderungen im Lastenheft verbalisiert hat, prüft der Autragnehmer, in der Definitionsphase, die Anforderungen und

beschreibt im Pflichtenheft, wie diese zu realisieren sind. Anschließend werden, in der objektorientierten Analyse, die verbalisierten Anforderungen in einem UML-Diagramm visualisiert.

5.3.1 Das Pflichtenheft

> Das Pflichtenheft enthält das vom Auftragnehmer erarbeitete Realisierungsvorhaben auf Grundlage des Lastenhefts. In ihm werden die Anwendervorgaben detailiert und in einer Erweiterung die Realisierungsanforderungen unter Berücksichtigung konkreter Lösungsansätze beschrieben. Dabei wird definiert „Wie“ und „Womit“ die Anforderungen zu realisieren sind.[30]

Das vollständige Pflichtenheft ist im Anhang unter Punkt A zu finden. Untergliedert ist es in die folgenden Abschnitte:

1. Die **Zielbestimmungen** erläutern die Ziele, die durch das zu erstellende Produkt erreicht werden sollen. Dabei ist der Punkt untergliedert in Muss-, Wunsch, und Abgrenzungskritereien. In den Musskriterien werden alle Anforderungen benannt, die unabdingbar sind. Die Wunschkritererien beschreiben die Anforderungen, die zwar wünschenswert, aber nicht Musskriterien sind. Die Abgrenzungkriterien verdeutlichen die Ziele, die mit dem zu erstellenden Produkt keinesfalls zu erreichen sind.

2. Der **Produkteinsatz** gibt Auskunft über den Bereich, in dem das zu erstellende Produkt Anwendung findet, an wen es sich richtet und unter welchen Bedingungen es eingesetzt wird.

3. In der **Produktübersicht** werden, durch den Einsatz von UML-Diagrammen, die Produktfunktionen visualisiert.

4. Die **Produktfunktionen** erweitern und konkretisieren die im Lastenheft benannten Funktionen.

5. Die von der Anwendung verwendeten **Produktdaten** werden beschrieben.

6. Die im Lastenheft genannten **Produktleistungen** werden unter diesem Punkt aufgeführt.

7. **Qualitätsanforderungen** Ziel dieses Kapitels ist es, die Qualitätsmerkmale an das zu entwickelnde Produkt zu erfassen.

[30] Vgl.:[ITHa06] Seite 335

8. Im Kapitel **Benutzeroberfläche** werden die Anforderungen an eine Benutzungsoberfläche gestellt, dazu zählen auch die ersten Oberflächenstudien.

9. Die **Nichtfunktionalen Anforderungen** beschreiben alle Anforderungen, die nicht einen der vorher genannten Punkten zuzuordnen sind. Dies können unter anderem zu beachtende Gesetze sein.

10. Das Kapitel **Technische Produktumgebung** beschreibt in den Punkten Software, Hardware, Orgware und Produktschnittstellen die Anforderungen an die technische Umgebung des Produktes.

11. **Spezielle Anforderungen an die Entwicklungsumgebung** beschreiben die technische Umgebung des Produktes während der Entwicklung, dabei bedient es sich einer Untergliederung wie der Punkt Technische Produktumgebung.

12. Die **Gliederung in Teilprodukte** beinhaltet alle getrennt zu entwickelnden Teilprodukte. Diese sind im Fall des OPC-OR-Mappers die Konfiguration, der Datentransfer und die Programmoberfläche.[31]

5.3.2 Die objektorientierte Analyse

Das Ziel der objektorientierten Analyse ist die im Pflichtenheft verbalisierten Anforderungen mit Hilfe von objektorientierten Konzepten zu modellieren. Es werden die Anforderungen und Wünsche des Auftraggebers an das Softwaresystem ermittelt und diese in ein Fachkonzept umgesetzt. Dabei gilt es zu beachten, dass alle Implementierungsaspekte auszuklammern sind. Es wird davon ausgegangen, dass perfekte Technik zum Einsatz kommt (Speichergröße und Verarbeitungszeit), auch die eventuelle Verteilung des Systems und die Form der Datenspeicherung wird zunächst nicht betrachtet.[32] Die daraus entstehende Spezifikation besteht aus einem statischen und einem dynamischen Modell. Welches dieser Modelle das größere Gewicht besitzt, ist vom zu erstellenden Produkt abhängig. Während das statische Modell bei Datenbankanwendungen wichtig ist, wird bei interaktiven oder technischen Systemen das dynamische Modell Verwendung finden.[33]

Die Analyse nach Paketen

Wie im Pflichtenheft beschrieben, ist die Entwicklung des zu erstellenden Produktes in die beiden Pakete Framework und Programmoberfläche zu teilen. Somit stehen die

[31] Vgl.:[HeBa00] Seite 117
[32] Vgl.:[WSSWT10]
[33] Vgl.:[HeBa00] Seite 378

ersten beiden Pakete für die Entwicklung schon fest. Um die Modularisierung zu gewärleisten, ist das Framework weiter aufzuspalten. Dabei werden zunächst, auf Grund der Austauschbarkeit, die Connectoren in extra Pakete gepackt. Dadurch entsteht ein Paket für den OPC-Connector und ein Paket für den DB-Connector. Aufgrund der getrennten Funktionen von Konfiguration und Datentransfer ist eine Paketisierung vorzuziehen. Daraus sind die im folgenden Diagramm visualisierten Pakete entstanden.

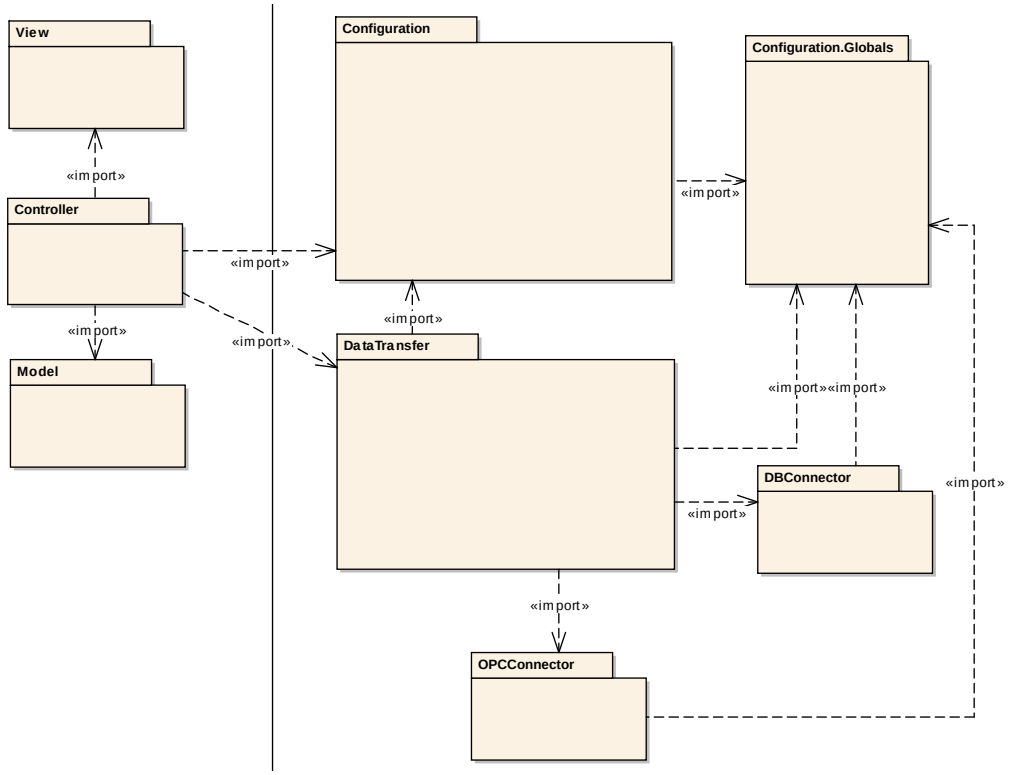

Abbildung 5.3: Die Gesamtarchitektur als Packagediagramm

Die Identifizierung von Klassen und Assoziationen

Bei der Identifizierung von Klassen und den Assoziationen sind die Produktfunktionen des Pfichtenhefts und das UseCase Diagramm heranzuziehen. Daraus werden Anhand der zuvor identifizierten Pakete die notwendigen Klassen und deren Beziehungen abgeleitet und deren Beziehung Zunächst werden die Connectoren mit den notwendigen Funktionen bedacht. Daraufhin folgt die Konfiguration, der Datentransfer und zum Schluss die Benutzungsoberfläche.

Um Verbindungen anzulegen, zu ändern bzw. zu löschen (Beschrieben in den Funktionen /PF020/, /PF030/, /PF050/ und /PF060/) werden die Entity-Klassen DbConnection

und OpcConnection angelegt. Diese enthalten die für die Verbindung relevanten Daten und sehen demzufolge exemplarisch für den DbConnector wie folgt aus:

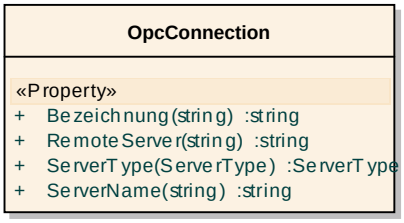

Abbildung 5.4: Die OpcConnection in der objektorientierten Analyse

Diese Connection-Klassen werden entweder als Parameter im Konstruktor oder an die Properties der Connectoren übergeben. Die Connectoren müssen zusätzlich zu dieser Property Methoden bereitstellen, um eine Verbindung herzustellen bzw. zu trennen. Diese Funktionen sind im Pflichtenheft unter den Produkfunktionen /PF010/ (Verbindung zur Datenbank herstellen) und /PF040/ (Verbindung zum Opc-Server herstellen) beschrieben. Erweitert werden die Klassen beim OPC-Server durch das Laden aller Items, das Laden eines Items mit Hilfe des Namens und der Methode, um das Feld SignalsValid zurückzusetzten.

Die Klasse des DBConnector muss zu den benannten Grundfunktionen Methoden bereitstellen, die alle Tabellen bzw. Procedurennamen sowie deren Spalten bzw. Parameternamen als Liste liefern. Um die aus dem OPC-Server gelesenen Daten in die Datenbanken schreiben zu können (beschrieben in der Funktion /PF170/), muss eine Methode implementiert werden, die dies ausführt.

Daraus ergibt sich für die Connectoren das folgende Klassendiagramm:

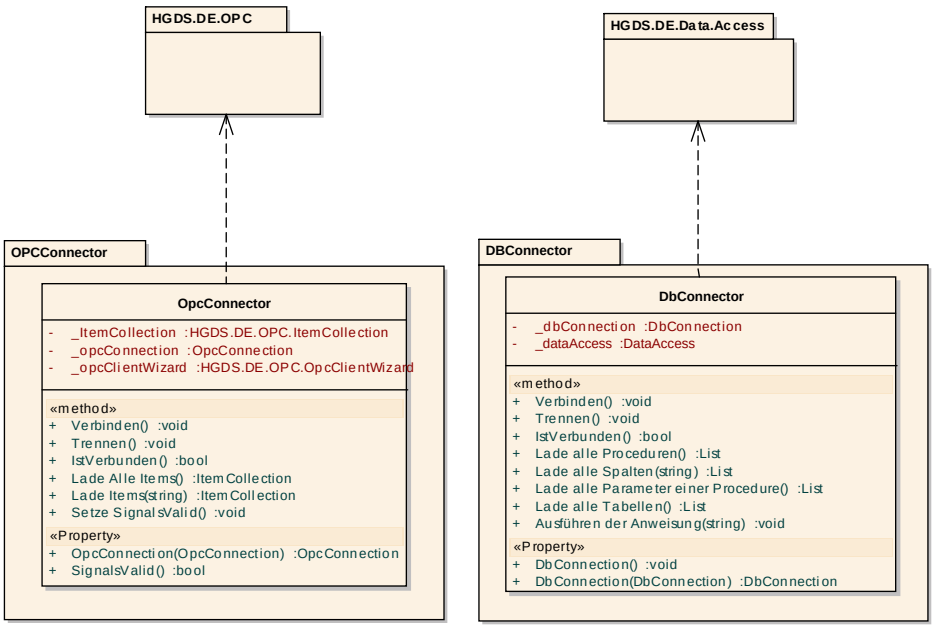

Abbildung 5.5: Die Connectoren in der objektorientierten Analyse

In der Configuration werden alle für das Mapping notwendigen Daten gehalten. Dazu gehören die OpcConnection die DbConnection sowie die zu mappenden Items. Unter diesem Namen muss eine Entity-Klasse implementiert werden die die dafür notwendigen Informationen bereitstellt.

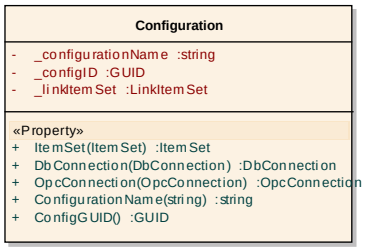

Abbildung 5.6: Die Configuration in der objektorientierten Analyse

Da der OPC-OR-Mapper mehrere Configuration und Connections verwaltetl, bedarf es einer Manager Klasse, die zunächst die Datenobjekte hält und weiterhin Funktionen bereitstellt, um diese zu anzulegen, zu bearbeiten und zu löschen. Die beschriebenen Datenobjekte werden dabei in generischen Listen gehalten, auf denen sich die bekannten Listenfunktionen ausführen lassen.

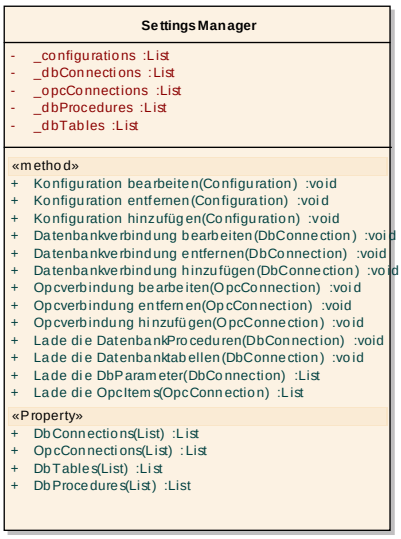

Abbildung 5.7: Der Settingsmanager in der objektorientierten Analyse

In der Configuration wird eine Liste namens LinkItemSet gehalten. Diese Klasse, welche von der generischen Klasse `List<>` erbt, enthält die eigentlichen Items. Das Item enthält dabei die Namen von Quelle und Ziel (Tabellenname und Spaltenname bzw. Prozedur und Parametername) sowie den Typ des Ziels (also Prozedur oder Tabelle). Weiterhin sind Felder für einen Zeitstempel und letztlich den Wert zu ergänzen. Daraus ergibt sich das Klassendiagramm des `LinkItemSet` und des `LinkItem` wie folgt:

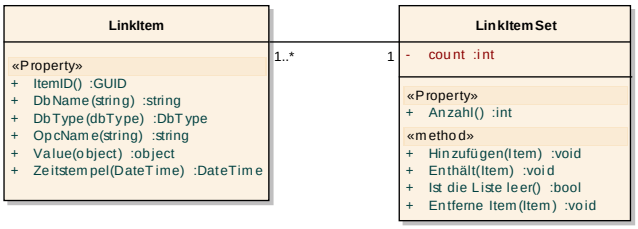

Abbildung 5.8: Das LinkItem und das LinkItemSet in der objektorientierten Analyse

Das eigentliche Mappen der Anwendung findet im `LinkItemManager` statt. Er stellt Methoden zum Auswählen, Abwählen und zur Verknüpfung von OPC-Item mit der Datenbankfunktion bereit. Diese benannten Funktionen sind im Pflichtenheft unter den Produktfunktionen /PF070/, /PF080/ und /PF090/ zu finden.

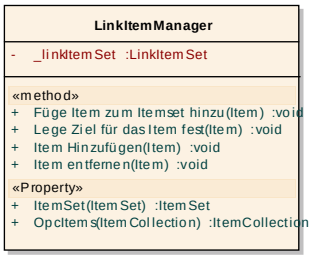

Abbildung 5.9: Der LinkItemManager in der objektorientierten Analyse

Abschließend können die identifizierten Klassen des Paketes Configuration diesem hinzugefügt und mit den Assoziationen versehen werden. Daraus ergibt sich das folgende Gesamtbild:

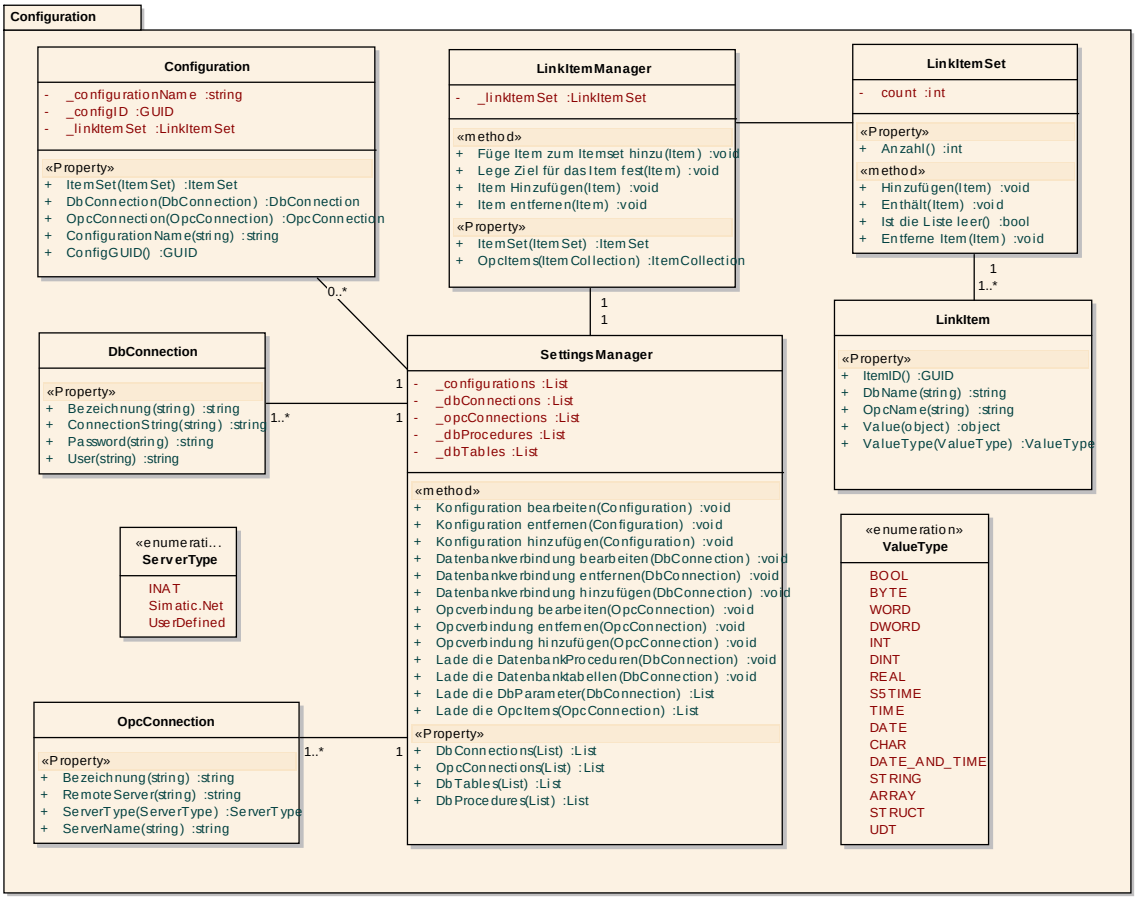

Abbildung 5.10: Das Paket Configuration in der objektorientierten Analyse

Die Configuration bildet die Grundlage für den Datenaustausch. Der DataTransferManager verwaltet eine Liste der angelegten Configurationen und stellt Methoden bereit, um einen Datentransfer zu starten bzw. zu stoppen. Dazu wird eine Klasse DataTransfer implementiert. Dieser Klasse wird eine Configuration übergeben und lässt sich anschließend starten. Dabei werden zunächst die Verbindung zum OPC-Server und zur Datenbank hergestellt. Anschließend wird ein Object vom Typ Converter erstellt, das die, in den Produktfunktionen beschriebenen, Funktionen /PF150/, /PF160/ und /PF170/ umsetzt. Dazu sind die Methoden `Lade OPCItems`, `Konvertieren`, `Erstelle Abfrage` und `Abfrage ausführen` notwendig. Das Klassendiagramm zum Konverter sieht also wie folgt aus.

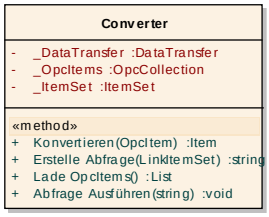

Abbildung 5.11: Das Converter in der objektorientierten Analyse

Dabei integrieren sich der `Converter`, der `DataTransfer` und der `DataTransferManager` in der objektorientierten Analyse wie folgt in das Klassendiagramm des Pakets DataTransfer.

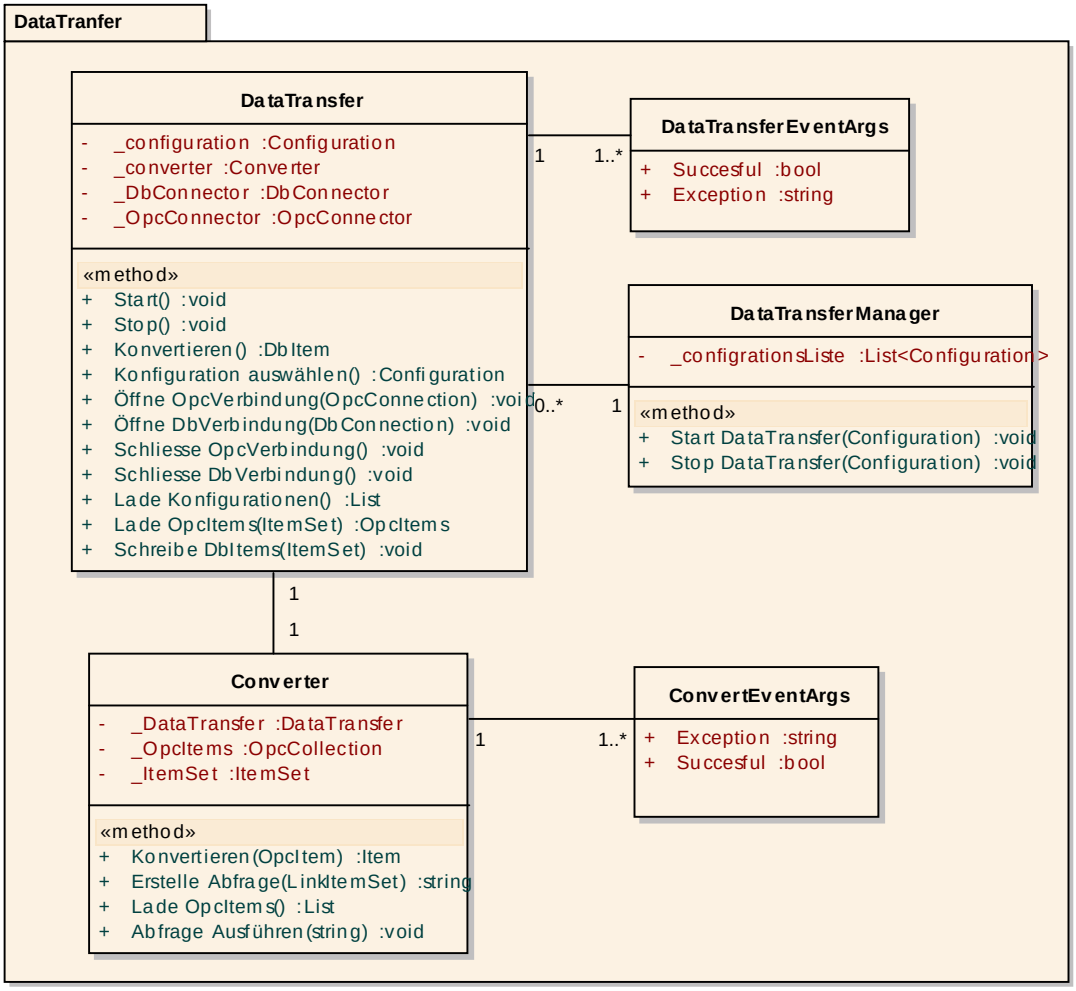

Abbildung 5.12: Das Paket DataTransfer in der objektorientierten Analyse

Die Oberfläche teilt sich in die Mainform und die Dialoge. Um die Bedingung der Austauschbarkeit einzuhalten, ist das Paket nach dem MVC-Prinzip (Modell View Control) zu implementieren. Dabei entsteht das anschließende Klassendiagramm.

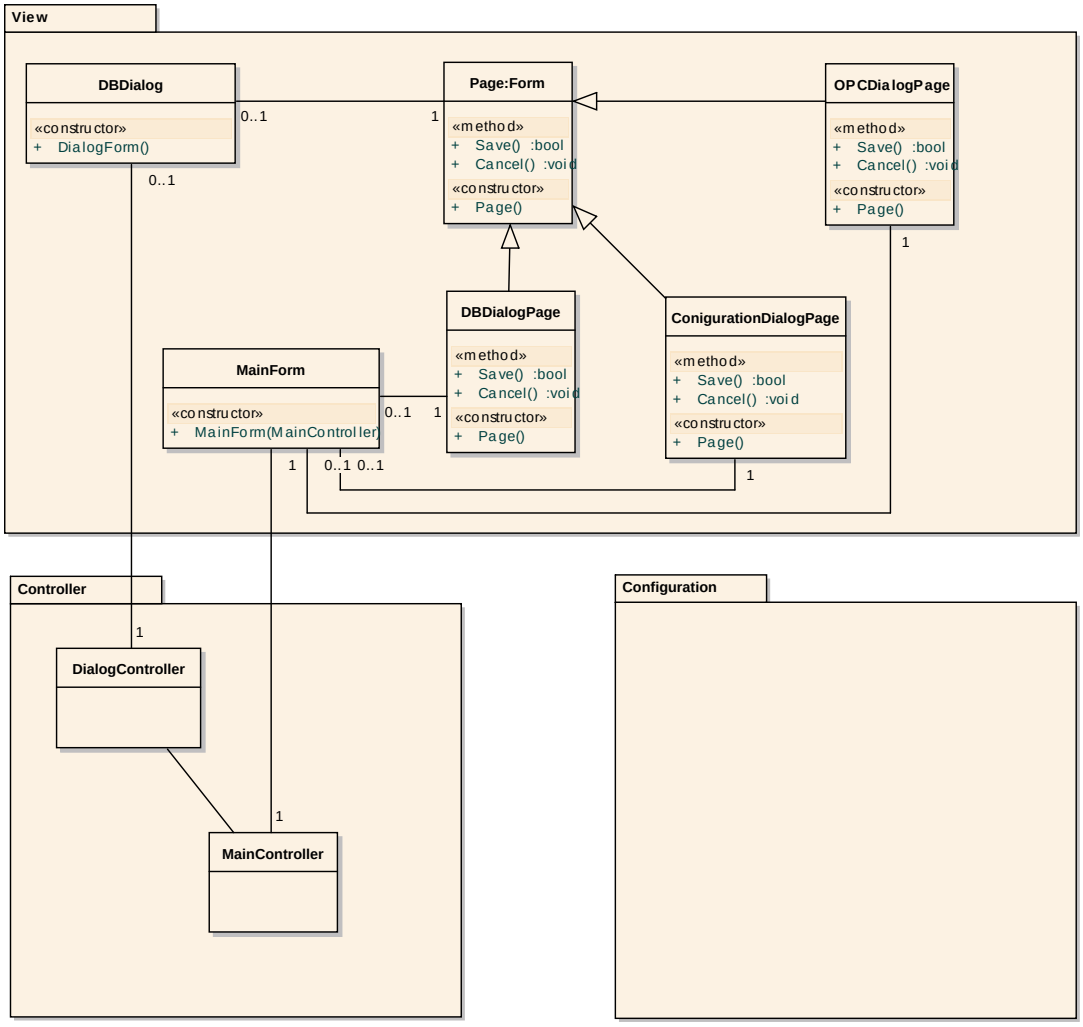

Abbildung 5.13: Das Frontend in der objektorientierten Analyse

5.4 Die Erweiterung der Definition im Entwurf

> Ziel der Entwurfsphase ist es, ausgehend von einer Produkt-Definition einen Produkt-Entwurf zu erstellen, der die Produkt-Anforderungen realisiert und die Anwednung architektonisch in die Anwendungs- und Plattformumgebung einbettet.[34]

Das Resultat der Entwurfsphase ist der Systementwurf. Er liefert dabei alle notwendigen Einflussfaktoren und Umgebungsbedingungen sowie das, an diese Faktoren und Bedingungen, angepasste Klassendiagramm.

5.4.1 Einflussfaktoren

Einen Teil der Einflussfaktoren, Randbedingungen und nichtfunktionalen Anforderungen sind schon im Pflichtenheft benannt, somit gilt dies als Zusammenfassung bzw. Erweiterung.

Einsatzbedingung

Das Produkt wird nach der Installation in einer Büroumgebung eingesetzt. Der Datentransfer wird für einen unbeaufsichtigten Dauerbetrieb ausgelegt. Das bedeutet auch, dass die Datenbank und der OPCServer sowie die SPS im Dauerbetrieb laufen. Der OPC-Server und die hier beschriebene Schnittstelle sind auf einem Zielsystem installiert.

Umgebungs- und Randbedingungen

Software

- Betriebssystem: Windows 2000 oder höher
- .NET Framework 2.0

Hardware

- Prozessor 400Mhz oder höher
- RAM 128 MB oder höher
- Speicherplatz 100 MB

[34] [HeBa00] Seite 986

Orgware

- Eine Verbindung zu einem Datenbankmanagementsystem muss vorhanden sein.

Entwicklungsumgebung

- Das Produkt wird mit der Entwicklungsumgebung Visual Studio 2005 unter Verwendung der Programmiersprache C-Sharp erstellt.

Produktschnittstellen

- Die Benutzer, die für den Zugriff auf das Datenbankmanagementsystem vorgesehen sind, müssen über die notwendigen Rechte verfügen, um alle Transaktion durchführen zu dürfen.

nichtfunktionale Produkt- und Qualitätsanforderungen

- Die Anwendung verfügt über eine leicht verständliche Programmoberfläche.
- Um die Erweiterbarkeit zu gewährleisten, ist die Anwendung modular zu gestalten.
- Das System ist innerhalb eines Tages von einem Fachmann installierbar.

5.4.2 Entscheidung zur Datenhaltung

Alle zu persistierenden Daten werden in XML-Files serialisiert. Dazu zählen die in der Analyse beschriebenen Connection-Klassen und die Configuration-Klasse.

5.4.3 Das Objektorientierte Design

Helmut Balzert beschreibt in seinem Lehrbuch der Softwartechnik[35] den Übergang von der objektorientierten Analyse(OOA) zum objektorientierten Design(OOD) als beste Lösung für den Übergang von der Definitions- in die Entwurfsphase. Beide Konzepte basieren auf den objektorientierten Konzepten, wodurch ein Strukturbruch vermieden wird.

Während in der OOA von idealen Systemumgebungen ausgegangen, nur die fachliche Lösung beschrieben wird, geht es im OOD um die technische Lösung unter Berücksichtigung der zuvor beschriebenen technischen Randbedingungen und der verwendeten

[35] [HeBa00] Seite 986

Plattform. Als Vorlage dient das in der Analyse erstellte Klassendiagramm, dass nun an die Umgebungsbedingungen und Einflussfaktoren angepasst werden muss. Da dieser Entwurf die Grundlage zur folgenden Implementierung liefert, gilt es auch die Syntax der verwendeten Programmiersprache zu beachten. Während bei der Modellierung in der Definitionsphase die deutsch Sprache angewendet wird, wird in der Entwurfsphase auf die englische Sprache zurückgegriffen. Weiterhin wird beobachtet, dass die Klassendiagramme nun auch Schnittstellen enthalten, die in der OOA noch nicht betrachtet wurden. Aus diesen Verfeinerungen entstehen die folgenden Klassendiagramme.[36]

Das Paket der Configuration wurde zusätzlich mit der Klasse DataTypeMatch versehen. Diese Klasse vergleicht die Datentypen des Opc-Servers mit den Datentypen der Datenbank. Auf die Funktion dieser Klasse wird im Kapitel 5.6 näher eingegangen.

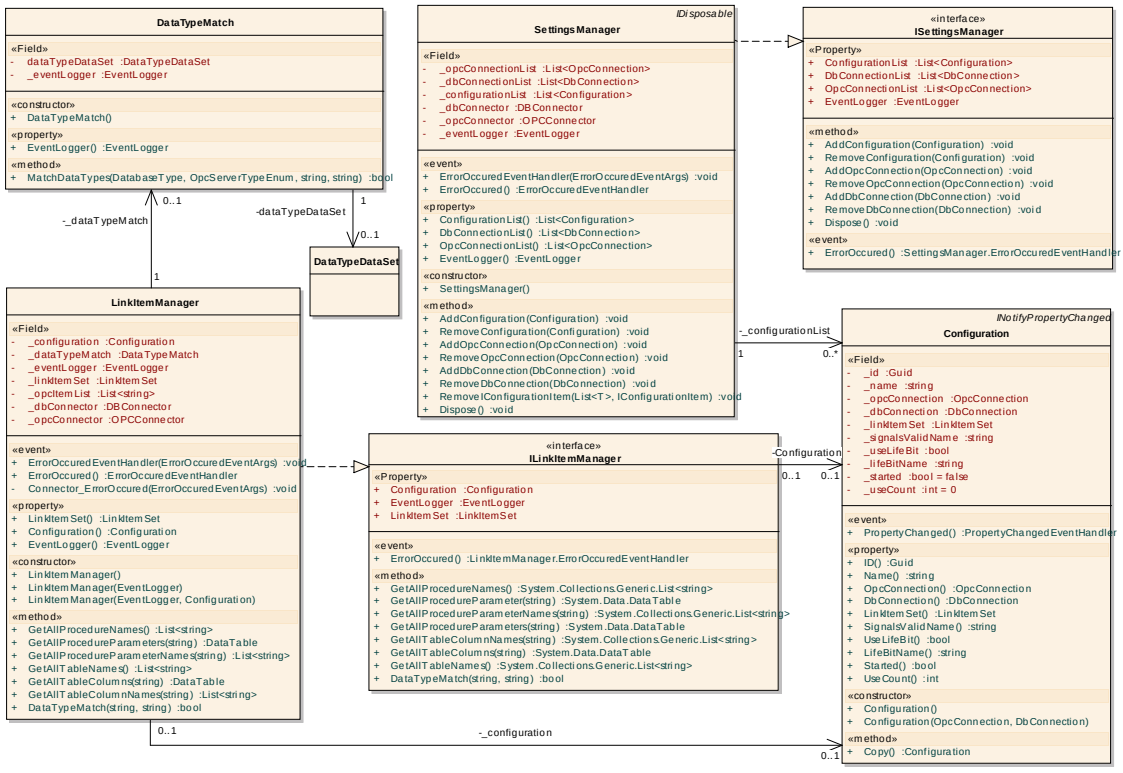

Abbildung 5.14: Klassendiagramm der Konfiguration

Wie bereits im Klassendiagramm der objektorientierten Analyse angedeutet, wurde das Paket `Configuration.Globals` zum Projekt hinzugefügt. Dieses Paket ist notwendig um „Zirkuläre Abhängigkeiten“ zu vermeiden. Diese Abhängigkeiten können entstehen, wenn eine Klasse A des Paketes X von eine Klasse B des Paketes Y und Klasse

[36] Die Klassendiagramme für die Connectoren und das GUI befinden sich im Anhang.

B wiederum von Klasse A abhängig ist. Um dies zu vermeiden, wurden die betreffenden Klassen in das Paket Configuration.Globals verschoben.

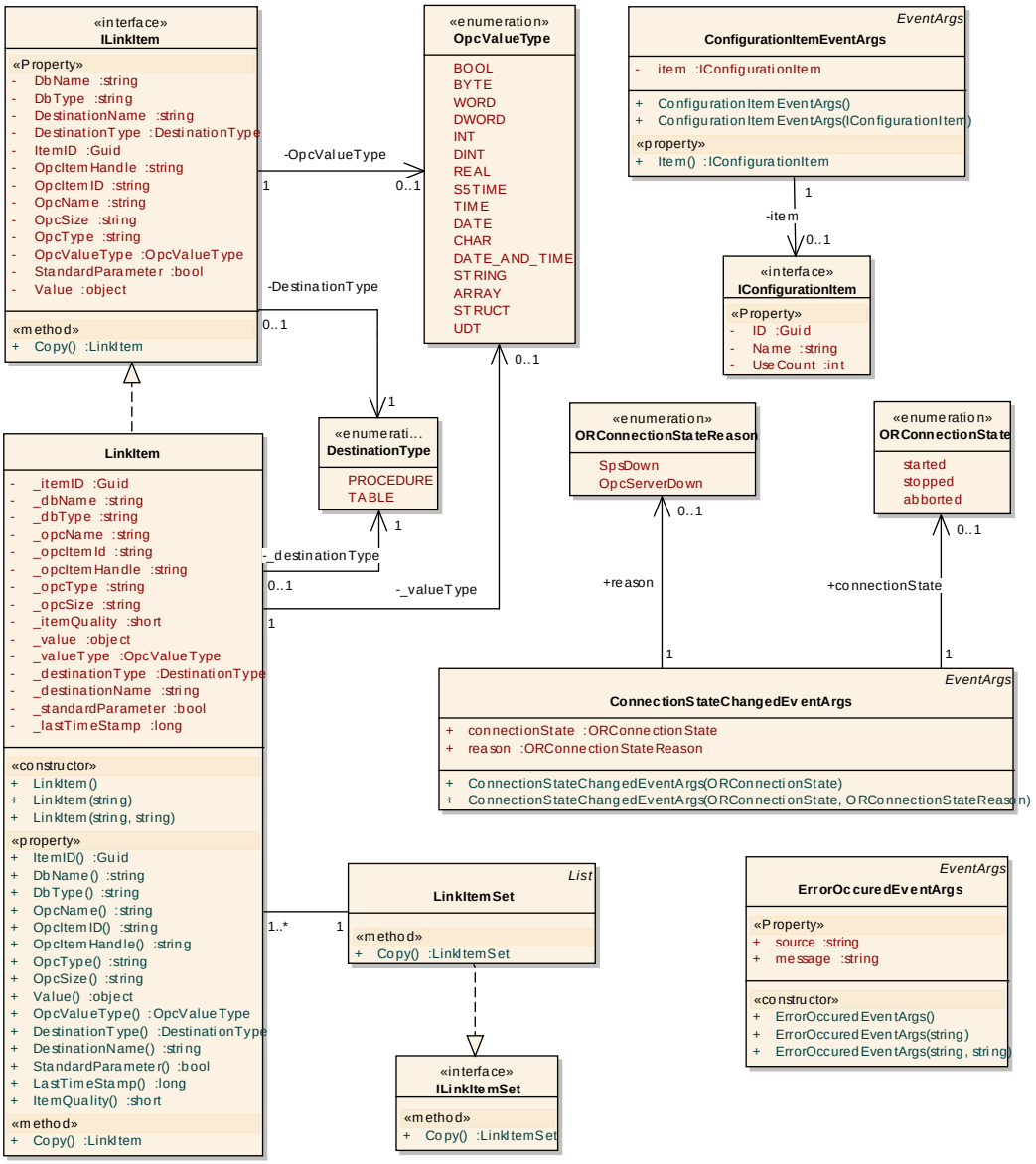

Abbildung 5.15: Klassendiagramm der Konfiguration Globals

Das Herstellen der Verbindungen wurde, gegenüber der objektorientierten Analyse, direkt an die Klasse Converter übergeben, um dem Gesetz von Demeter[37] gerecht zu werden. Dieses Gesetz besagt, daß Objekte nur mit Objekten in ihrer unmittelbaren Umgebung kommunizieren sollen. Das Paket wurde ebenfalls um die Klasse

[37] Vgl.:[MSLD11]

`OpcXmlFileCreator` erweitert. Die Aufgabe dieser Klasse besteht darin, ein Konfigurationsfile für den OpcConnector anhand der Configuration zu erstelllen.

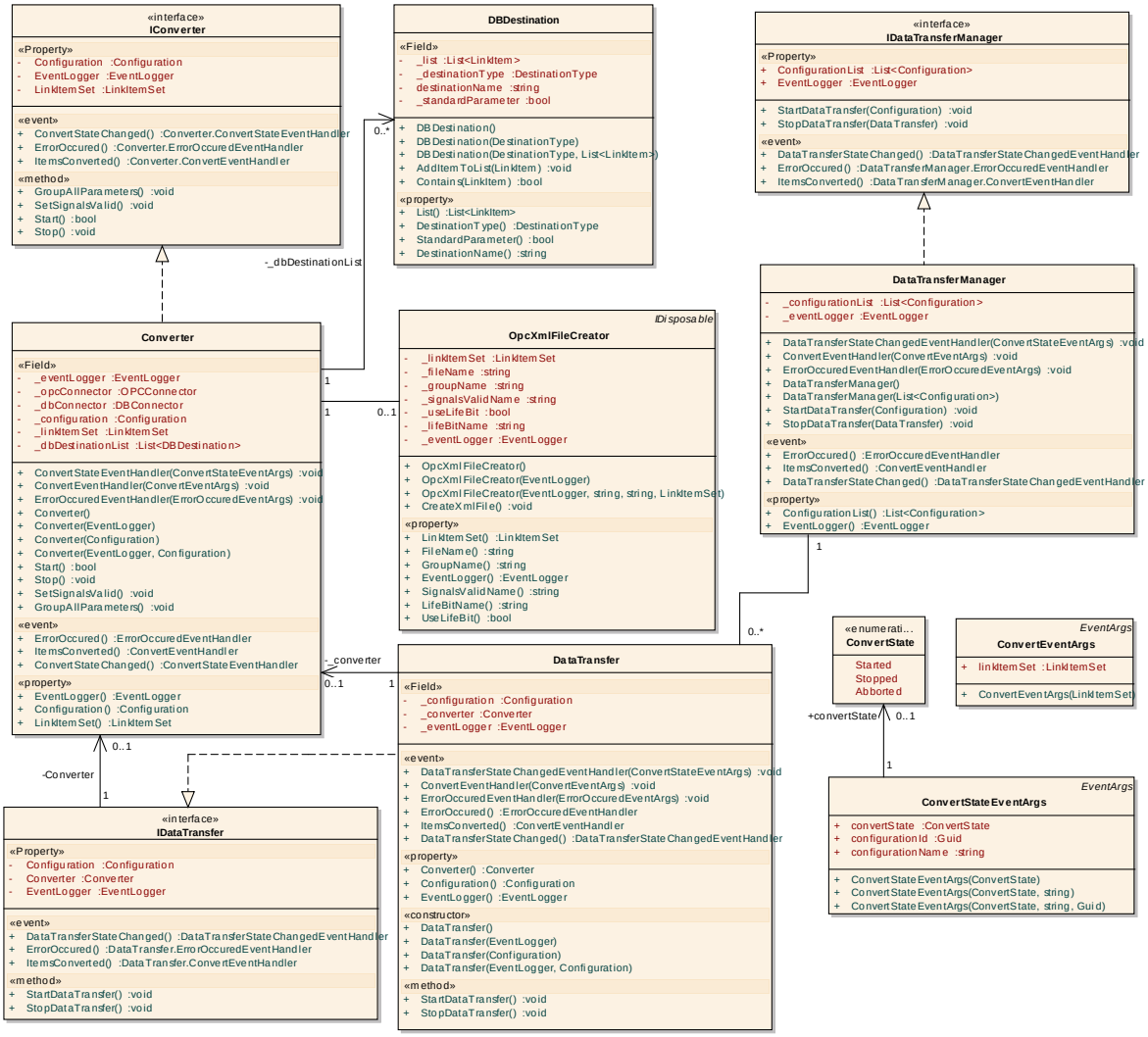

Abbildung 5.16: Klassendiagramm des Datentransfers

5.5 Die technische Realisierung in der Implementierung

In der Implementierung werden die in den vorhergehenden Phasen entworfenen Systemkomponenten technisch realisiert. Dabei ist die Software-Architektur so ausgelegt, dass der zu implementierende Umfang einer Komponente überschaubar bleibt und von einer bzw. wenigen Personen durchgeführt werden kann.

Dabei gilt es, nach Helmut Balzert[38] die folgenden 4 Prinzipien zu beachten:

- Prinzip der Verbalisierung
- Prinzip der programmadäquaten Datentypen
- Prinzip der Verfeinerung
- Prinzip der integrierten Dokumentation

Prinzip der Verbalisierung

Zu diesem Prinzip gehören die aussagekräftige und mnemonische Namensgebung, geeignete Komentare und die selbstdokumentierende Programmiersprache. Damit soll eine gute Lesbarkeit des Quellcodes erreicht werden.

Prinzip der programmadäquaten Datentypen

Die problemadäquaten Datentypen sind die bewusste Widerspiegelung der fachlichen Datentypen bzw. -strukturen als Atribute bzw. Parameter von Operationen. Auch die Wahl der Datentypen ist Bestandteil einer selbstdokumentierenden Programmiersprache.

Prinzip der Verfeinerung

Die Verfeinerung bezieht sich auf das strukturierte Anlegen von Methoden und Klassen. Dabei werden komplexe Operationen in einzelne Schritte aufgeteilt, die durch Kommentar beschrieben sind. Dadurch sollen Algorithmen durchschaubarer gemacht werden.

Prinzip der integrierten Dokumentation

Zu diesem Prinzip wird eine kurze Beschreibung des Programms in den Quellcode integriert. Zusätzlich muss der Quellcode noch dokumentiert werden. Visual Studio bietet hier Funktionen, die es möglich machen, eine Entwicklerdokumentation zu erstellen.

[38] Vgl.: [HeBa00] Seite 1069

Ein weiteres Prinzip, dass Anwendung finden sollte, ist die Wiederverwendbarkeit, die sich auch schon bei der Wahl der Datenzugriffsschicht für die Datenbank bzw. den Opc-Server als sehr wichtig herausgestellt hat.

Unter Anwendung dieser Prinzipien soll das Framework nun entwickelt werden. Dabei wird nachfolgend auf Besonderheiten während der Implementierung eingegangen.

Zur Implementierung werden die im Entwurf erstellten Klassendiagramme und die Sequenzdiagramme herangezogen. Sollten während der Entwicklung Änderungen am Modell der objektorientierten Analyse vorgenommen werden, so sind diese auch in das jeweilige Diagramm zu übertragen.

Entwickelt wird das System, wie im Entwurf[39] angekündigt, unter Verwendung Microsofts Visual Studio 2005 mit dem Framework 2.0 und der Programmiersprache C-Sharp.

5.6 Besonderheiten während der Implementierung

Der Vergleich der Datentypen

Wie im objektorientierten Design herausgearbeitet, soll der Vergleich der Datentypen als eine Besonderheit während der Implementierung erläutert werden.[40] OPC Server wie auch die relationalen Datenbanken haben eine Reihe von Datentypen, die nicht identisch sind. Beim Anlegen einer Konfiguration ist deshalb darauf zu achten, dass die beiden Datentypen zueinander passen, denn während das Schreiben eines `int` in einen `varchar[50]`-Feld problemlos auszuführen sein sollte, führt das Schreiben eines `string` in ein `int`-Feld mit Sicherheit zu einer Exception. Um dieses Problem zu lösen, wird eine Klasse namens `DataTypeMatch` implementiert. Diese Klasse wird ein DataSet vom Typ DataTypeDataSet instanzieren, das die folgende Struktur hat:

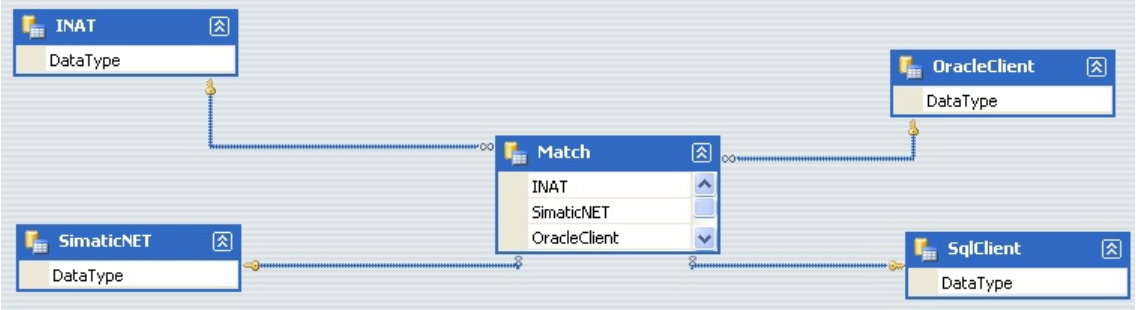

Abbildung 5.17: Das DataTypeDataSet

[39] Kapitel: 5.4.1
[40] Weitere Besonderheiten sind dem Projektordner auf der beiliegenden CD zu entnehmen.

Die Tabellen INAT, SimaticNET, OracleClient und SqlClient dienen dabei der Eineindeutigkeit, es wird somit verhindert, dass Datentypen doppelt angelegt werden. Die Tabelle Match wird für den eigentlichen Vergleich verwendet und sieht in der Rasteransicht exemplarisch so aus:

Daten für Match

	INAT	SimaticNET	OracleClient	SqlClient
▶	int	int	integer	int
	(NULL)	char	char	char
	string	string	char	char
	dint	dint	integer	int
	(NULL)	date_and_tim	(NULL)	datetime
	bit	bool	(NULL)	bit
*				

Abbildung 5.18: Das DataTypeDataSet in der Datenraster-Ansicht

Dabei ist für jeden verwendeten OPC Server und auch Datenbank eine eigene Spalte vorhanden. In dieser Spalte werden die jeweiligen Datentypen einander zugewiesen. Im folgenden Listing wird die Methode MatchDataTypes abgebilldet. Der Methode werden dabei Servertyp und Datenbanktyp sowie die Namen der beiden Datentypen übergeben. Aus den übergebenen Namen der Datentypen wird eine DataView mit Hilfe einer Abfrage erstellt. Abschließend wird geprüft ob die DataView Reihen enthält (bei Übereinstimmung) oder nicht. Das Ergebnis dieser Prüfung als Bool ist der Rückgabewert der Methode.

Listing 5.1: Die MatchDataTypes-Methode

```
1  public bool MatchDataTypes(DatabaseType dataBaseType,
2                  OpcServerTypeEnum opcServerType, string sDataBaseType,
3                  string sOpcType)
4  {
5      DataTable dtMatch = dataTypeDataSet.Match;
6      try
7      {
8          DataView dv = new DataView(dtMatch, dataBaseType.ToString()
9                      + "='" + sDataBaseType.ToLower() + "' AND " +
10                                     opcServerType.ToString() + "='"
11                     + sOpcType.ToLower()+"'",
12                                     string.Empty,
13                     DataViewRowState.CurrentRows);
14         return dv.Count > 0;
15     }
16     catch(Exception ex)
17     {
18         this._eventLogger.WriteEvent("DataTypeMatch",
19             EventMessage.UnhandledException, ex,
20             EventID.UnhandledException);
21         return false;
```

```
22      }
23  }
```

Das Arbeiten mit den Objekten

Bei Objekten handelt es sich um Referenztypen. Da mit Hilfe der Referenz direkt auf dem Objekt versehentlich Änderungen vorgenommen werden können, empfielt es sich diese Objecte vorher zu replizieren. Auf diesen Replikationen kann gearbeitet und nach Bestätigung der Änderung das Original mit der Replikation überschrieben werden bzw. die Replikation verworfen werden. Dies betrifft in diesem Fall vor allem die Entity-Klassen OpcConnecion, DbConnection, LinkItem und Configuration. Während es für die Klassen OpcConnection DbConnection und LinkItem ausreichend ist, eine flache Copie zu erstellen, enthalten die Klassen Configuration und LinkItemSet wiederum Objekte die eine flache Kopie ausschließen. Es muss also eine Methode implementiert werden, die eine tiefe Kopie der Configuration erstellt. Dazu wird ein rekursives Vorgehen gewählt, daß anhand des Beispiels LinkItemSets erklärt wird. Das LinkItem bedient sich der von der Klasse `Object` geerbten Methode `MemberwiseClone()`.

Listing 5.2: Die Methode um das LinkItem zu clonen

```
1 public LinkItem Copy()
2         {
3             return (LinkItem)this.MemberwiseClone();
4         }
```

Die `Copy()` der Configuration ruft dabei die Methode `Copy()` des LinkItemSets auf, die wiederum die Methode `Copy()` Methode des LinkItemSets aufruft.

Listing 5.3: Die Methode um das LinkItemSet zu clonen

```
1  public LinkItemSet Copy()
2         {
3             LinkItemSet copiedLinkItemSet = new LinkItemSet();
4             foreach (LinkItem linkItem in this)
5             {
6                 copiedLinkItemSet.Add((LinkItem)linkItem.Copy());
7             }
8             return copiedLinkItemSet;
9         }
```

Dadruch wird gleichzeitig erläutert, warum die Klasse LinkItemSet implementiert wird. Sie erbt ihre Funktion von der generischen Basisklasse `List<LinkItem>` und wird ergänzt um die `Copy()`-Methode, die, wie beschrieben, jedes im Set enthaltene Item copiert.

Der Eventlogger

In einer Anwendung können Fehler auftreten, auf die reagiert werden muss. Das Programm muß dabei trotz des Fehlers stabil weiterarbeiten können. Gleichzeitig sollten die Fehler aber festgehalten werden. Die Firma HGDS GmbH verwendet ein selbstentwickeltes Paket namens EventLogger. Sinn und Zweck dieses Paketes ist folgender: Wenn ein Fehler aufgetreten ist, wird die Fehlermeldung mit der Ursache im Eventlog des Computers festgehalten. Der Vorteil dieser Methode ist darin zu sehen, daß der Anwender nicht mit überladenen Fehlermeldungen überfahren wird, denn er erfährt nur, das ein Fehler aufgetreten ist, während der `EventLogger` den Fehler in das Eventlog des Computers schreibt. So ist es für die Techniker der HGDS GmbH leichter möglich Fehler nachzuvollziehen.

5.7 Das Testen im Softwarelebenszyklus

Um eine qualitativ hochwertige Software zu erstellen und den geforderten Qualitätsbedingungen gerecht zu werden, ist das Testen notwendig. Dabei kann, wie dem Vorgehensmodell zu entnehmen ist, zwischen den fünf folgenden Testzyklen unterschieden werden:

- Klassentest
- Modultest
- Integrationstest
- Systemtest
- Abnahmetest

Der Systemtest soll im Rahmen dieser Arbeit aber das letzte angewendete Testverfahren sein, da der Abnahmetest in der Regel auf dem Kundensystem und durch den Kunden durchgeführt wird.

5.7.1 Einsatz des Testframeworks NUnit

Für das .NET Framework existieren verschiedenste Testframeworks, eines der bekanntesten ist dabei das NUnit-Testframework, daß auch hier Verwendung findet. Es orientiert sich dabei an den xUnit-Konzept, das vauch Kent Beck vertritt.[41] Vorteile beim Einsatz eines Testframeworks sind:

- das Testframework kann in die Entwicklungsumgebung Visual Studio 2005 integriert werden.
- bei jedem Testlauf wird jede zuvor erstellte Testklasse bzw. jedes Modul erneut getestet, wodurch ungewollte Seiteneffekte, die während der weiteren Implementierung auftreten können, aufgedeckt werden.

Beim Einsatz dieses Testverfahrens ist die Strategie des „Test First“ anzuwenden, das besagt, daß vor der implementierung der eigentlichen Klasse die Testklasse zu implementieren ist.[42]

5.7.2 Klassentest

Der Klassentest ist eine Besonderheit bei diesem angepassten Vorgehensmodell und wird mit Hilfe des Testframeworks durchgeführt. Dazu wird dem Projekt ein neues Projekt vom Typ Klassenbibliothek hinzugefügt. Um in diesem Projekt die Funktionen des

[41] Vgl.:[StPi10]
[42] Das erstellte Testpackage ist auf der beiliegenden CD zu finden.

Testframeworks nutzen zu können, muss ein Verweis auf das Framework hinzugefügt werden. Anschließend muss jede dll, die getestet werden soll, dem Projektordner hinzugefügt werden. Nun bekommt jede Klasse ein Pendant als Testklasse, in dieser Klasse muss zu jeder erstellten Methode eine Testmethode implementiert werden. Alle Testklassen müssen mit `[TestFixture]` und Methoden mit `[Test]` markiert werden. Als Ergebnis ist die korrekte Funktion der Klassen anzusehen.

5.7.3 Modultest

Nachdem alle Klassen eines Systems getestet wurden, schließt sich der Modultest an. Das Ziel dieses Tests ist, wie auch beim Klassentest, die fachliche Korrektheit der Teilergebnisse. Kennzeichnend für den Modultest ist, daß nur ein Softwarebaustein, isoliert von den anderen Bausteinen, getestet wird. Durch die Isolierung können komponentenexterne Einflüsse ausgeschlossen und somit bei der Fehlersuche ausgegrenzt werden.

5.7.4 Integrationstest

In dieser Stufe wird das fehlerfreie Zusammenarbeiten von voneinander abhängigen Komponenten überprüft. Das hier anzuwendende Testverfahren nennt sich Whitebox-Verfahren. Die Tests werden mit Kenntnis von der inneren Funktionsweise durchgeführt. Als Beispiel lässt sich das Kontrollflussorientierte Testverfahren benennen, dabei gilt es sicherzustellen, dass die Testfälle in Bezug auf die Überdeckung des erstellten Quellcodes Hinlänglichkeiten erfüllen, dies wird anhand der folgenden Kriterien bemessen. Die Ziele der einzelnen Kriterien sind dabei:

- beim **Anweisungsüberdeckungstest**, daß jede Codezeile des Programms mindestens einmal ausgeführt wird.
- beim **Zweigüberdeckungstest** sollen nicht ausführbare Zweige des Programms aufgespürt werden.
- beim **Pfadüberdeckungstest** werden die Pfade durch das Modul betrachtet.
- das beim **Bedingungsüberdeckungstest** jede atomare Bedingung einmal den Wert true und den Wert false annimmt.

Als Minimalkriterium ist dabei der Zweigüberdeckungstest anzusehen. Der hier neben dem Anweisungsüberdeckungstest, der durch die Verwendung des Testframeworks gegeben ist, ebenfalls Verwendung finden soll. Dabei gilt es zu beachten, dass selbst wenn alle nicht ausführbare Zweige gefunden wurden, diese trotzdem ausführbar sein können. Die Codebereiche der Exceptionbehandlung sind ein Beispiel dafür. Auch für Schleifen ist dieses Testverfahren nicht geeignet, da diese nicht ausreichend durchlaufen werden.

5.7.5 Systemtest

Nach dem abgeschlossenen Interationstest folgt der Systemtest. In diesem Testzyklus wird das Gesamtsystem getestet, dabei wird das Blackboxverfahren angewendet. Das bedeutet, dass im Gegensatz zum zuvor benannten Whitebox-Testverfahren nicht der Quellcode für die Tests relevant ist, sondern vielmehr das Verhalten der Software an sich. Während in den vorigen Testzyklen aus der Sicht des Entwicklers getestet wurde, soll nun aus der Sicht des Kunden bzw. Anwenders getestet werden. Es kann zwischen zwei Testzielen unterschieden werden, die auf unterschiedliche Art und Weise getestet werden müssen. Den funktionalen Anforderungen und den nicht funktionalen Anforderungen.

Die **funktionalen Anforderungen** beschreiben, was das System bzw. Teilsystem leisten soll. Sie sind unabdingbar, da das Produkt sonst nicht einzusetzen ist. Merkmale dafür sind laut der ISO 9126 die Angemessenheit, Richtigkeit, Interoperabilität, Ordnungsmäßigkeit und Sicherheit.[43] Die Anforderungen werden in der Analysephase gesammelt und im Lastenheft bzw. Plichtenheft festgehalten. Beim Testen kann zwischen dem anforderungsbasierten Testen und dem anwendungsfallbasierten Testen unterschieden werden, dazu werden die niedergeschriebenen Anforderungen bzw. UseCase Beschreibungen verwendet.

In den **nicht funktionalen Anforderungen** wird die Qualität der Software beschrieben, die mit der Kundenzufriedenheit einhergeht. Merkmale dafür sind laut ISO 9126 die Zuverlässigkeit, die Benutzbarkeit und die Effizienz.[44] Als Testverfahren sind hierbei verschiedene Lasttests und die Tests der zuvor benannten Merkmale zu benennen. Bei diesen Testzielen werden auch die Programm- und die Benutzerdokumentation auf die Übereinstimmung mit dem erstellten Produkt überprüft.

5.7.6 Abnahmetest

Der Abnahmetest ist wie der Systemtest ein Test des Gesamtsystems, nur mit dem Unterschied, dass diesmal beim Kunden und eventuell auch durch den Kunden getestet wird. In diesem Testzyklus steht die Sicht und das Urteil des Kunden bzw. des Anwenders im Vordergrund. Dabei geht es vor allem darum, dass die Software auch wirklich, die im Pflichtenheft, zugesicherten Funktionen liefert. Wenn der Auftraggeber und der Anwender eines Softwaresystems unterschiedliche Personen sind, so ist dieser Test durch beide Personen durchzuführen. Viele Auftraggeber machen die Rechnungsstellung von den bestandenen Abnahmetests abhängig. Dabei sollte der Abnahmetest keinesfalls den Systemtest ersetzen, erst wenn der Systemtest erfolgreich absolviert wurde, sollte das Produkt zum Kunden gehen.

[43] Vgl.:[ASTL04] Seite 57
[44] Vgl.:[ASTL04] Seite 61

5.7.7 Testfälle

Der Testfall beschreibt einen funktionalen Softwaretest, einer benannten Komponente, unter Anwendung einer Testmethode. Für jeden Testfall muss eine Ausgangssituation (Vorbedingung) und ein erwartetes Ergebnis bzw. Verhalten beschrieben werden. Ebenfalls sollten alle Randbedingungen, die für den Test gelten und einzuhalten sind festgehalten werden.[45] Die Testfälle lassen sich nach den folgenden beiden Kriterien unterscheiden:

- Testfälle zur Prüfung der erwarteten Reaktionen bzw. Ergebnisse, unter gültigen Bedingungen, werden Positivtests genannt.
- Testfälle, in denen gewollt ein Fehler, bei den Eingabedaten bzw. den Randbedingungen, herbeigeführt wird, um die Ausnahmebehandlung zu prüfen werden Negativtests benannt.

Als Beispiel für einen Positivtest dient der Testfall in der Tabelle 5.7.

5.7.8 Testumgebung

Unabhängig vom Rechner, auf dem die Anwendung installiert wird, kommen die folgenden Datenbanken zum Einsatz:

- Microsoft SQL Server 2008 R2
- Oracle 10 Release 2

[45] Vgl.:[ASTL04] Seite 21

Die Anwendung wurde im Labor der Firma HGDS GmbH auf zwei virtuellen Maschinen mit unterschiedlichen OPC-Servern getestet.

Virtuelle Maschine 1:

Betriebssystem:	Microsoft Windows Server 2003 Standard Edition SP2
Prozessor:	Intel(R) Xeon(R) CPU E5310 1,60 Ghz 512 MB RAM
OPC-Server:	INAT OPC-Server TCPIPH1 Version 4.0.11.00

Tabelle 5.5: Testumgebung Rechner 1

Virtuelle Maschine 2:

Betriebssystem:	Microsoft Windows XP Professional Version 2002 SP3
Prozessor:	Intel(R) Xeon(R) CPU E5310 1,60 Ghz 512 MB RAM
OPC-Server:	Industrial Ethernet SOFTNET-S7-Lean Version 7.1 OPC-Server

Tabelle 5.6: Testumgebung Rechner 2

5.7.9 Testergebnisse

Das Testergebnis ist das Resultat des durch den Testfall beschriebenen Vorgehen Testergebnisse liefern nur der Integrationstest, Systemtest und der Abnahmetest. Die Ergebnisse zu den Klassen- und Modultests sind wie beschrieben die Funktion der Klassen bzw. Modultests. Dazu wurde ein Paket erstellt, das alle Testklassenklassen enthält.[46]

Vorraussetzung für einen Testergebnis ist der Testfall. Als Beispiel dient der folgende Testfall[47]:

[46] Das Paket findet sich auf der beiliegenden CD im Projektverzeichnis.
[47] Weitere Testfälle und Testergebnisse sind der beiliegenden CD zu entnehmen.

Pos	Bedingung	Beschreibung
1	Beschreibung	Es soll eine neue Datenbankverbindung angelegt werden!
2	Vorbedingung	Eine Datenbankverbindung mit diesem Namen existiert nicht.
3	Randbedingungen	Keine
4	Testdaten	DatabaseType: SQLClient Bezeichnung: Datenbankverbindung 1 Connectionstring: Data Source=192.168.1.3; Initial Catalog=NORTHWIND; User Id=sa; Password=; Timeout: 15 Sekunden
4	Erwartetes Ergebniss	Die Datenbankverbindung wurde erfolgreich angelegt.
5	Tatsächliches Ergebnis	Die Datenbank wurde unter der Bezeichnung Datenbankverbindung 1 angeleget
6	Testergebnis	Bestanden
7	Bemerkungen	Keine

Tabelle 5.7: Der Testfall Datenbankverbindung anlegen

Die in den Testzyklen aufgedeckten Probleme machen eventuell eine Änderung der Testpriorisierung notwendig. Korrigierte Fehler erfordern zusätzliche Fehlernachtests, bei denen es auch gilt, die eventuell auftretenden Seiteneffekte nicht außer Acht zu lassen. Es kann auch passieren, dass zusätzliche Tests notwendig werden, weil Probleme nicht vollständig rerproduzierbar bzw. analysierbar sind. Gerade unter Betrachtung dieses Punktes ist der Einsatz des Testframeworks ein effektiver Lösungsansatz.

6 Zusammenfassung und Ausblick

6.1 Ergebnisse

Als Ergebnis der Entwicklung und nach Durchführung der Tests konnte eine lauffähige Version an die HGDS-GmbH übergeben werden. Dabei wurden alle Anforderungen, die an das Produkt gestellt wurden, erfüllt. Das Produkt wird nach dem Abnahmetest in den produktiven Einsatz übernommen.

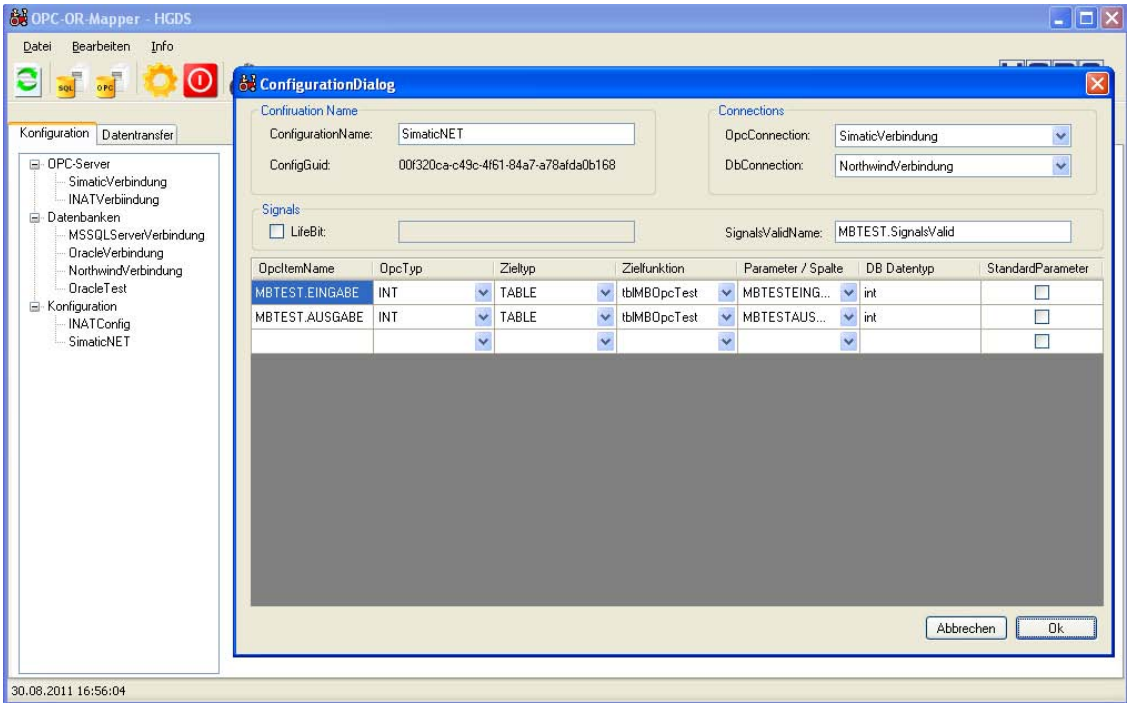

Abbildung 6.1: Das Hauptfenster des OPC-OR-Mapper

In einer Demonstration wurden die Anwendungsfunktionen dargestellt. Dazu gehörte das Anlegen von Verbindungen und Konfigurationen. Die Datenübertragung wurde von einem SimaticNET OPC-Server zu einer MSSQL-Datenbank durchgeführt. Dabei wurden, wie der Abbildung 6.1 zu entnehmen ist zwei Items angelegt.

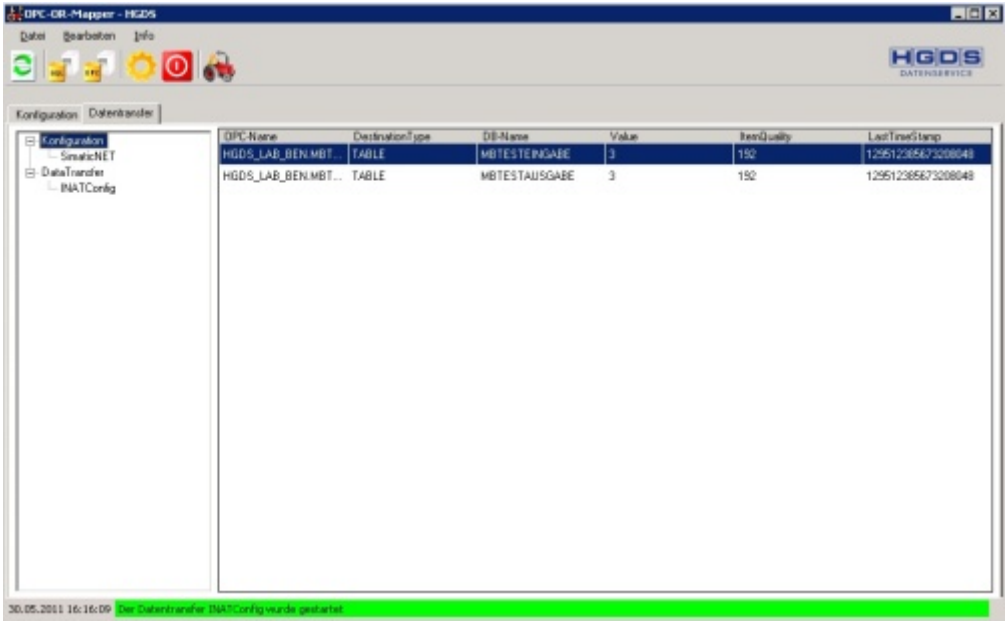

Abbildung 6.2: Der OPC-OR-Mapper in Aktion

Nach Abschluß der Konfiguration wurde der Datentransfer gestartet und die Werte damit zyklisch in die Datenbank geschrieben.

Zusammenfassend wird eingeschätzt, dass unter Anwendung des beschriebenen Vorgehens ein verwendbares Resultat erzielt wurde.

6.2 Reflektion des Erreichten

Die Aufgabe bestand in der Schaffung eines Frameworks, das einen konfigurierbaren Datenaustausch zwischen Automatisierungsanlagen und Datenbanken ermöglicht. Um diesen Datenaustausch ohne jeglichen Programmieraufwand konfigurieren zu können, wurde dem Framework ein Frontend zur Seite gestellt, das diese Aufgabe erfüllt. Die Aufgabe wurde im Lösungskonzept unter Verwendung der im Kapitel 3 vorgestellten Technologien konzipiert und anschließend unter Anwendung eines angepassten V-Modells realisiert. Nach der Implementierung wurde das Produkt mit Hilfe der beschriebenen Testverfahren getestet. Diese Tests wurden bestanden, dennoch ist es möglich, dass während des Abnahmetests (der noch nicht ausgeführt wurde) und dem späteren Einsatz noch Fehler auftreten können. Auf diese Fehler muss in der Phase Wartung, die nicht Bestandteil der Aufgabe ist, reagiert werden und die Fehler sind zu beheben. Während der Entwicklung hat sich aber auch gezeigt, daß diese Aufgabe noch ein enormes Potential an Ausbaumöglichkeiten bietet. Darauf wird nachfolgend unter dem Punkt 6.3 eingegangen.

6.3 Ausblick

Im diesem Absatz werden die Möglichkeiten der Weiterentwicklung unter wissenschaftlicher und praktischer Hinsicht betrachtet.

6.3.1 Möglichkeiten der Weiterentwicklung

Möglichkeiten zur Weiterentwicklung bestehen bei den folgenden Punkten:

- In erster Linie steht hierbei der Einsatz der Unified Architecture, um die Unabhängigkeit von Betriebssystemen zu gewärleisten. Dies wäre aber auch im Hinblick auf die Konfiguration der DCOM Schnittstelle ein großer Fortschritt.
- Mit einer Portierung der Anwendung in das Framework 3.5 (oder höher) wäre auch der Einsatz des von Microsoft angebotenen „LINQ“ denkbar.
- Es sollten weitere Provider für die Datenbanken eingebunden werden, durch die verwendete Datenzugriffschicht ist dies mit einem relativ geringem Aufwand denkbar.
- Derzeit unterstützt das Framework den INAT- und den SimaticNet OPC-Server. Diese Produktpalette sollte erweitert werden.

6.3.2 Praktischer Einsatz

Unter praktischer Hinsicht,

- ist vor allem das Auslesen aller vorhandenen Items vom OPC-Server anzustreben. In der gegenwärtigen Version, smüssen die Items händisch erfasst werden, dies sollte automatisiert werden. Dabei ist nicht außer Acht zu lassen, dass es sich bei großen Automatisierungsanlagen um mehr als 1000 Items handeln kann. Um die Übersicht zu behalten, wäre eine Lösung mit einer Baumstruktur anzustreben. Dies würde in vielerlei Hinsicht Erleichterungen bringen, vor allem dahingehend, dass die Items zur Konfiguration nicht mehr bekannt sein müssen.
- Auch der Einsatz von den im Pflichtenheft, unter den Wunschkriterien, benannten Aktualisierungmethoden, wird einen erheblichen Vorteil in der Anwendbarkeit bringen.
- Damit der Datentransfer beim Schließen der Anwendung nicht beendet wird, sollte dieser als Dienst gestartet werden. Dabei ist wie auch in der jetzigen Version eine Überwachung der aktuellen Items anzustreben.

Literaturverzeichnis

[ASTL04] Spillner, Andreas; Linz, Tilo: *Basiswissen Softwaretest*. - 2. überarbeitete Auflage - Heidelberg: dpunkt.verlag GmbH, 2004

[CJDa03] Date, C.J.: *An Introduction to Database Systems*. 8. Auflage - Silicon Valley: Pearson Addison, 2003

[Codd70] Codd Edgar F.: *A Relational Model of Data for Large Shared Data Banks*Communikations of the ACM. 1970

[HeBa00] Balzert Helmut: *Lehrbuch der Software-Technik*. 2. Auflage - : Spektrum Akademischer Verlag GmbH, 2000

[ITHa06] Hübscher; Petersen; Rathgebe; Richter; Scharf: *IT-Handbuch*. 4. Auflage - Braunschweig: Bildungshaus Schulbuchverlage Westermann Schrodel Diesterweg Schöningh Winklers GmbH, 2006

[KiBa07] King, Gavin; Bauer, Christian: *Java Persistence mit Hibernate* . Auflage :Carl Hanser Fachbuchverlag, 2007

[ViCS10] Kühnel, Andreas: *Visual C# 2010* . 5. Auflage : Galileo Computing, 2010

[CHHG06] Huschto, Christian: *Konzipierung und Implementierung einer Datenzugriffsschicht für den Einsatz von verschiedenen Daten-Providern in mehrschichtigen Anwendungen für das Microsoft .NET Framework V1.1.* - 2006 - Mittweida, HS-Mittweida, Fakultät Mathematik Naturwissenschaften Informatik, Diplomarbeit, 2006

[MaBe11] Benndorf, Maik: *Praktikumsbericht: Von der SPS in das DBMS*. - 2011 - Mittweida, HS-Mittweida, Fakultät Mathematik Naturwissenschaften Informatik, Praktikumsbericht, 2011

[TUCb04] Schönefeld, Frank: *Einführung in Datenbanken*. - 2004 - Cottbus, TU-Cottbus Fakultät Informatik, Vorlesung 2004

[WSSWT10] Schubert, Wilfried: *Vorlesung Softwaretechnik*. - 2010 - Mittweida, HS-Mittweida, Fakultät Mathematik Naturwissenschaften Informatik, Vorlesung, 2010

[Advo11] Advosol Inc.: *Exchange Server*. URL:<`http://www.advosol.com/pc-26-5-exchange-server.aspx`>, abgerufen am 19.07.2011

[Asco11] ascolab GmbH: *OPC UA Übersicht*. URL:<http://www.ascolab.com/de/unified-architecture/uebersicht.html>, abgerufen am 27.07.2011

[AyRa07] Eini, Oren: *implementing-linq-for-nhibernate-a-how-to-guide*. URL:<http://ayende.com/blog/2227/implementing-linq-for-nhibernate-a-how-to-guide-part-1>, abgerufen am 24.07.2011

[BRAl10] Bayrischer Rundfunk: *Grundkurs Mathematik (7) 7.1. Der Relationsbegriff*. URL:<http://www.br-online.de/br-alpha/grundkurs-mathematik/grundkurs-mathematik-7-DID1248340493424/grundkurs-mathematik-mathematik-relationsbegriff-ID1248353176876.xml>, abgerufen am 04.08.2011

[CoIn11] Mühsig, Robert: *HowTo: O/R Mapper LINQ to SQL – Einführung & einfaches manuelles Mapping*. URL:<http://code-inside.de/blog/2008/01/15/howto-or-mapper-linq-to-sql-einfhrung-einfaches-manuelles-mapping/>, abgerufen am 15.07.2011

[FraHo11] Meier, Matthias: *Softwareentwicklungsprozess*. URL:<http://www.ipa.fraunhofer.de/index.php?id=308>, abgerufen am 27.07.2011

[JBos09] Patricio, Anthony: *NHibernate for .NET*. URL:<http://community.jboss.org/wiki/NHibernateForNET>, abgerufen am 17.07.2011

[JInt10] Intrinsyc Software International, Inc.: *Configuring DCOM for Remote Access*. URL:<http://j-integra.intrinsyc.com/support/com/doc/remoteaccess.html>, abgerufen am 19.07.2011

[MiRa10] MidrangeMagazin: *Service Oriented Architecture*. URL:<http://www.midrangemagazin.de/mm/artikel.html?id=5864&SID=rrnmscaxf>, abgerufen am 20.07.2011

[MoSc10] Morgenroth, Karlheinz; Schmied, Jürgen: *Wann welches Vorgehensmodell Sinn macht*. URL:<http://www.elektronikpraxis.vogel.de/themen/embeddedsoftwareengineering/management/articles/273975/>, abgerufen am18.08.2011

[Msdn11] Microsoft Deutschland GmbH, *LINQ to SQL,*. URL:<http://msdn.microsoft.com/de-de/library/bb386976.aspx>, abgerufen am 14.07.2011

[MSLD11] Microsoft Deutschland GmbH: *Muster in der Praxis: Kohäsion und Kopplung* URL:<http://www.msdn.microsoft.com/de-de/magzine/cc947917.aspx#id0070040>, abgerufen am 28.08.2011

[NUni11] NUnit.org: *NUnit.* URL:<http://www.nunit.org/>, abgerufen am 23.08.2011

[OPCF11] OPC-Foundation: *About OPC*. URL:<http://www.opcfoundation.org/SiteMap.aspx?MID=AboutOPC>, abgerufen am 16.07.2011

[StPi10] Pichel, Stefan: *NUnit*. URL:<http://profi-software-entwicklung.de/index.php?option=com_content&view=article&id=9&Itemid=11>, abgerufen am 25.08.2011

[Tech10] TechTarget inc.: *object-relational mapping,*. URL:<http://searchwindevelopment.techtarget.com/definition/object-relational-mapping>, abgerufen am 15.07.2011

Anhang A: Das Pflichtenheft

Version	Stand	Änderung/ Ergänzung	Status
0.3	13.05.2011	Pflichtenheft	vorläufig
0.4	25.05.2011	Pflichtenheft	abgeschlossen

Tabelle A.1: Pflichtenheft Revisionstabelle

A.1 Zielbestimmungen

Bei dieser Aufgabe handelt es sich um eine Schnittstelle, die in der Lage ist mit einem zuvor gewählten OPC-Server und einer ebenfalls gewählten Datenbank zu kommunizieren. Dabei werden die zuvor gewählten Daten aus dem OPC-Server gelesen und nach der Verarbeitung in der Datenbank abgebildet. Diese Schnittstelle ist für die Konfiguration und das anstoßen des Datentransfers in eine GUI zu implementieren.

A.1.1 Musskriterien

Verwaltung der Datenbankverbindungen

- Hinzufügen einer neuen Verbindung
- Ändern einer bestehenden Verbindung
- Löschen einer bestehenden Verbindung

Verwaltung der OPC-Server-Verbindungen

- Hinzufügen einer neuen Verbindung
- Ändern einer bestehenden Verbindung
- Löschen einer bestehenden Verbindung

Verwaltung der Konfigurationen

- Anlegen einer neuen Konfiguration
  - Auswahl der OPC-Server-Verbindung
  - Auswahl der Datenbank-Verbindung
  - Auswahl der Quelldaten
  - Auswahl des Ziels (Stored Procedure oder Tabelle)
  - Zuordnung der Quelldaten zu den Zieldaten

- Ändern einer bestehenden Konfiguration
  - Auswahl der Quelldaten
  - Abwahl der Quelldaten
  - Änderung des Ziels (Stored Procedure oder Tabelle)
- Löschen einer bestehenden Konfiguration

Verwaltung Datenaustausch

- Starten des Datenaustauschs unter Verwendung der triggerbasierten Aktualisierungsmethode
- Beenden des Datenaustauschs

A.1.2 Wunschkriterien

Die folgenden Funktionalitäten werden angestrebt, aber nicht zugesichert:

- Erstellung eines Konfigurationsscripts für den OPC-Server
- die folgenden Aktualisierungsmethoden
  - Intervallbasierte Aktualisierung
  - Änderungsbasierte Aktualisierung
  - Grenzwertbasierte Aktualisierung
- eine Benutzeranmeldung und Rechteverwaltung
- die in den Musskriterien benannten Funktionen lassen sich mit einer Konsole ausführen

A.1.3 Abgrenzungskriterien

Die Software wird in dieser Ausbaustufe nicht in der Lage sein, die folgenden Funktionalitäten zu erfüllen:

- Es wird davon ausgegangen, dass die Zielfunktionen bzw. Tabellen schon existieren. Somit ist es nicht möglich Daten in ein nicht existentes Ziel zu mappen bzw. dieses mit Hilfe des Programms anzulegen.
- Bisher ist nur ein lesender Zugriff auf die Daten des OPC-Servers sowie schreibender Zugriff auf die Daten in der Datenbank angedacht. Es gilt jedoch zu beachten, dass die triggerbasierte Aktualisierung einen schreibenden Zugriff verlangt. Da in dieser das Feld „SignalsValid“ nach dem Lesen gesetzt werden muss um das fortschreiben des OPC-Servers zu ermöglichen. Andernfalls ist nur das Auslesen eines einzigen Datensatzes möglich.

A.2 Produkteinsatz

A.2.1 Anwendungsbereiche

Das Produkt kommt überall da zum Einsatz, wo es notwendig ist Daten einer Steuerung zyklisch zu persistieren.

A.2.2 Zielgruppen

Das Produkt dient der Firma HGDS bei der Einrichtung von prozessorientierten Schnittstellen sowie produzierenden Unternehmen bei der Betreibung dieser.

A.2.3 Betriebsbedingungen

Das Produkt wird nach der Installation in einer Büroumgebung eingesetzt. Der Datentransfer wird für einen unbeaufsichtigten Dauerbetrieb ausgelegt. Das bedeutet auch, dass die Datenbank und der OPC-Server sowie die SPS im Dauerbetrieb laufen. Der OPC-Server und die hier beschriebene Schnittstelle sind auf einem Zielsystem installiert.

A.3 Produktübersicht

Abbildung A.1: Erweitertes UseCase Produktübersicht

Aus der Gesamtübersicht ergeben sich die folgenden Teilübersichten.

A.3.1 Verbindung zur Datenbank verwalten

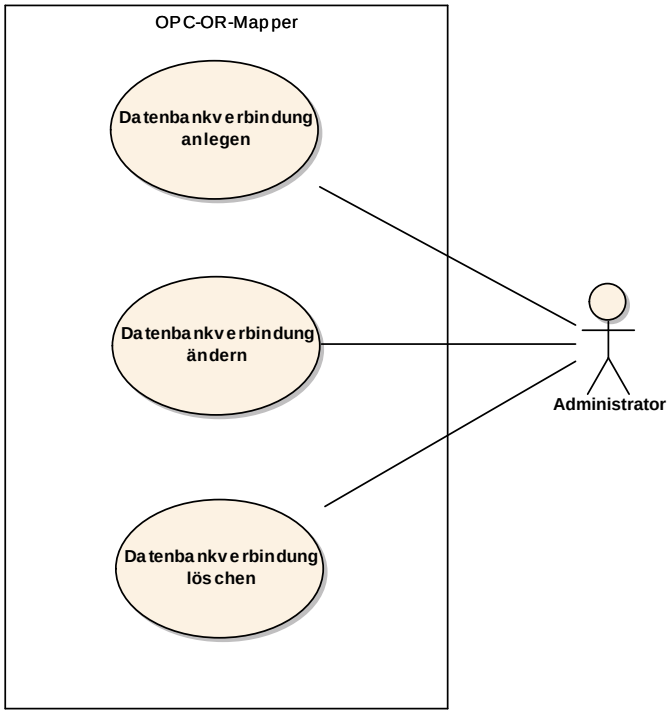

Abbildung A.2: Erweitertes UseCase Verbindung zur Datenbank verwalten

A.3.2 Verbindung zur OPC-Server verwalten

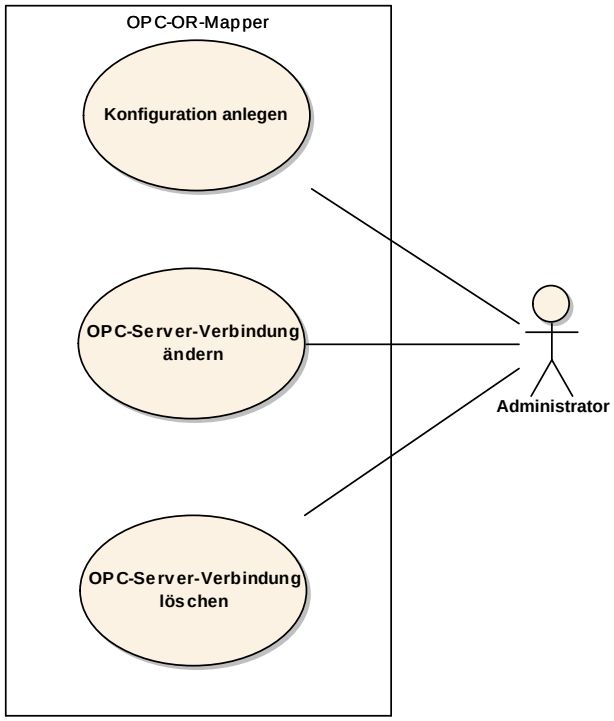

Abbildung A.3: Erweitertes UseCase Verbindung zum OPC-Server verwalten

A.3.3 Konfiguration verwalten

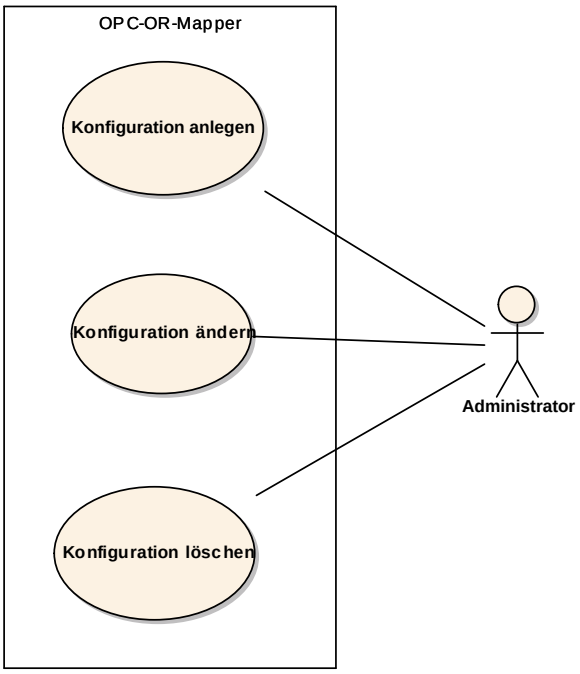

Abbildung A.4: Erweitertes UseCase Konfiguration verwalten

A.3.4 Verwaltung des Datentransfers

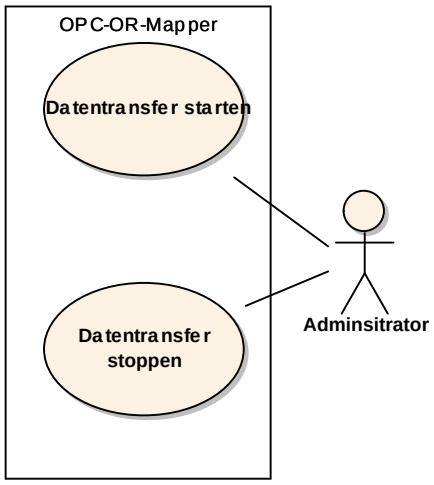

Abbildung A.5: Erweitertes UseCase Verwaltung des Datentransfers

A.3.5 Datenflussdiagramm

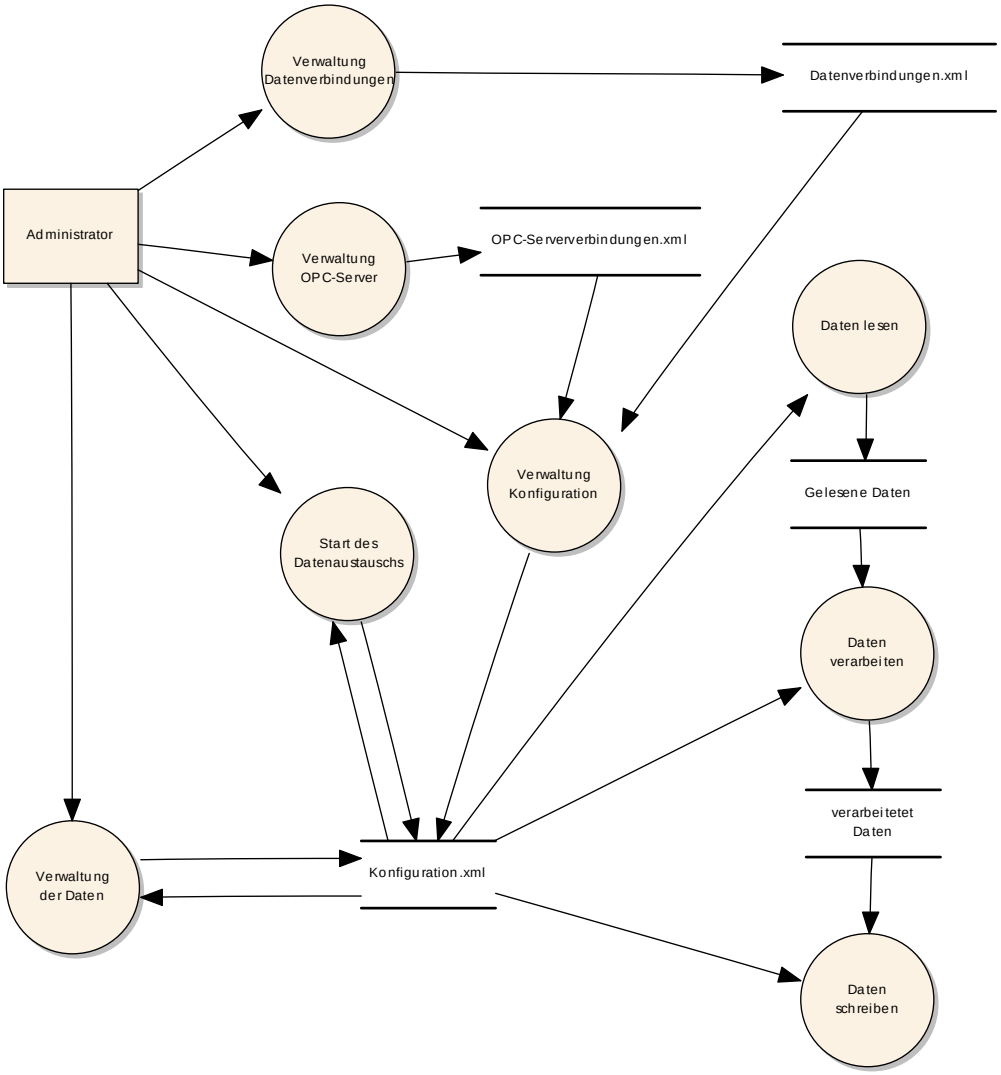

Abbildung A.6: Datenflussdiagramm

A.4 Produktfunktionen

Pos	Name	Kennung	Beschreibung
1	/PF010/	Funktion: Akteur: Beschreibung:	Verbindung zur Datenbank herstellen Administrator Anhand der Eingaben des Akteurs wird eine Verbindung zur Datenbank hergestellt. Hierzu werden die bereits vorhandenen Datenbankschnittstellenklassen verwendet.
2	/PF020/	Funktion: Akteur:	Verbindung zur Datenbank ändern Administrator

Pos	Name	Kennung	Beschreibung
		Beschreibung:	Eine bestehende Verbindung lässt sich ändern.
3	/PF030/	Funktion: Akteur: Beschreibung:	Verbindung zur Datenbank löschen Administrator Eine Verbindung zur Datenbank soll sich erst nach Rückfrage löschen lassen, da hierdurch der Verkehr zwischen OPC-Server und Datenbank beeinträchtigt wird.
4	/PF040/	Funktion: Akteur: Beschreibung:	Verbindung zum OPC-Server herstellen Administrator Aufbau einer Verbindung zu einem OPC-Server anhand der zu erfassenden Parameter.
5	/PF050/	Funktion: Akteur: Beschreibung:	Verbindung zum OPC-Server ändern Administrator Der Administrator ändert eine bestehende Verbindung zum OPC-Server.
6	/PF060/	Funktion: Akteur: Beschreibung:	Verbindung zum OPC-Server löschen Administrator Da sich das Löschen der Serververbindung auf die Kommunikation zwischen diesem und der Datenbank auswirkt, ist das Löschen zuvor zu bestätigen.
7	/PF070/	Funktion: Akteur: Beschreibung:	Verwaltung der zu mappenden Daten Administrator Die vom OPC-Server vorliegenden Daten können mit dieser Funktion aus- bzw. abgewählt werden.
8	/PF080/	Funktion: Akteur: Beschreibung:	Auswählen der abzubildenden Daten Administrator Alle Daten des OPC-Servers werden in einer Liste dargestellt und mittels aktivierten Checkboxen für das Abbilden übernommen.
9	/PF090/	Funktion: Akteur: Beschreibung:	Abwählen der abzubildenden Daten Administrator Alle Daten des OPC-Servers werden in einer Liste dargestellt und mittels deaktivierten Checkboxen nicht für das Abbilden übernommen.
10	/PF100/	Funktion: Akteur:	Erstellung einer Konfiguration Administrator

Pos	Name	Kennung	Beschreibung
		Beschreibung:	Auf der Grundlage der zuvor erstellten Verbindungen zur Datenbank und zum OPC-Server, der selektierten zu mappenden Daten ist es mit dieser Funktion möglich, eine Konfiguration zu erstellen.
11	/PF110/	Funktion: Akteur: Beschreibung:	Ändern einer Konfiguration Administrator Diese Funktion schließt die Funktionen Datenbankverbindung ändern, OPC-Serververbindung änder sowie die Verwaltung zu mappenden Daten ein.
12	/PF120/	Funktion: Akteur: Beschreibung:	Löschen einer Konfiguration Administrator Eine Konfiguration wird nach Bestätigung des Löschvorgangs entfernt.
13	/PF130/	Funktion: Akteur: Beschreibung:	Datenaustausch Starten Administrator Nachdem die Konfiguration erstellt wurde, wird unter Verwendung der triggerbasierte Aktualisierung der zyklische Datentransfer gestartet.
14	/PF140/	Funktion: Akteur: Beschreibung:	Datenaustausch Stoppen Administrator Eine Funktion die durch die Funktion /PF130/ gestartet ist wird hiermit beendet.
15	/PF150/	Funktion: Akteur: Beschreibung:	Daten aus der Steuerung lesen System Angestoßen durch das Starten des Datentransfers werden die konfigurierten Datenquellen mit Hilfe des OPC-Servers aus der Steuerung gelesen. Die Aktualisierung der Daten ist dabei abhängig von der im Datenaustausch gewählten Methode. Nach dem erfolgreichen Lesen der Daten des OPC-Servers wird das Feld „Signals_Valid“ wieder auf „0“ gesetzt.
16	/PF160/	Funktion: Akteur:	Daten verarbeiten System

Pos	Name	Kennung	Beschreibung
		Beschreibung:	In dieser Funktion finden die konfigurierten Anpassungen der ausgelesenen Werte (z.B. Typkonvertierungen) an die Struktur der Datenbank statt.
17	/PF170/	Funktion: Akteur: Beschreibung:	Daten in der Datenbank speichern System Nach der Verarbeitung der Daten werden diese, unter Berücksichtigung der in der Konfiguration festgelegten Verknüpfungen, in der Datenbank persistent abgelegt.

Tabelle A.2: Produktüberischt

A.5 Produktdaten

Pos	Name	Kennung	Beschreibung
1	/PD010/	Name der Datengruppe: Maximale Anzahl: Beschreibung:	Datenverbindung 100 Informationen zur Datenverbindung • Bezeichnung • GUID • Host • User • Passwort • Name der Datenbank • Typ der Datenbank
2	/PD020/	Name der Datengruppe: Maximale Anzahl: Beschreibung:	OPC-Server-Verbindungen 100 Informationen zur OPC-Server-Verbindung • Bezeichnung • GUID • Host • Typ des Servers
3	/PD030/	Name der Datengruppe: Maximale Anzahl: Beschreibung:	Konfiguration 15 Informationen zur Konfiguration • Bezeichnung • Datenbankverbindung • OPC-Server-Verbindung • Datum • Typ der Überwachung
4	/PD040/	Name der Datengruppe: Maximale Anzahl: Beschreibung:	Item 1.000.000 Informationen zum erfassten Item • Bezeichnung • GUID • OPC-Name • OPC-Datentyp • Wert • DB-Name • DB-Datentyp • TimeStamp

Tabelle A.3: Produktdaten

A.6 Produktleistungen

Pos	Name	Kennung	Beschreibung
1	/PL010/	Beschreibung:	Transaktionen nach dem ACID

Tabelle A.4: Produktleistungen

A.7 Qualitätsanforderungen

Pos	Name	Sehr gut	Gut	Normal	Nicht relevant
1	Funktionalität	X			
2	Angemessenheit		X		
3	Richtigkeit	X			
4	Interoperabilität	X			
5	Sicherheit		X		
6	Ornungsmäßigkeit	X			
7	Zuverlässigkeit	X			
8	Reife		X		
9	Fehlertroleranz	X			
10	Funktionalität	X			
11	Benutzbarkeit			X	
12	Vertsändlichkeit				X
13	Erlernbarkeit			X	
14	Bedienbarkeit			X	
15	Effizienz			X	
16	Zeitverhalten			X	
17	Verbauchsverhalten			X	
18	Änderbarkeit	X			
19	Analysierbarkeit		X		
20	Modifizierbarkeit	X			
21	Stabilität	X			
22	Prüfbarkeit	X			
23	Übertragbarkeit	X			
24	Anpassbarkeit	X			
25	Installierbarkeit		X		
26	Konformität	X			
27	Austauschbarkeit	X			

Tabelle A.5: Erweiterte Qualitätsanforderungen

A.8 Benutzeroberfläche

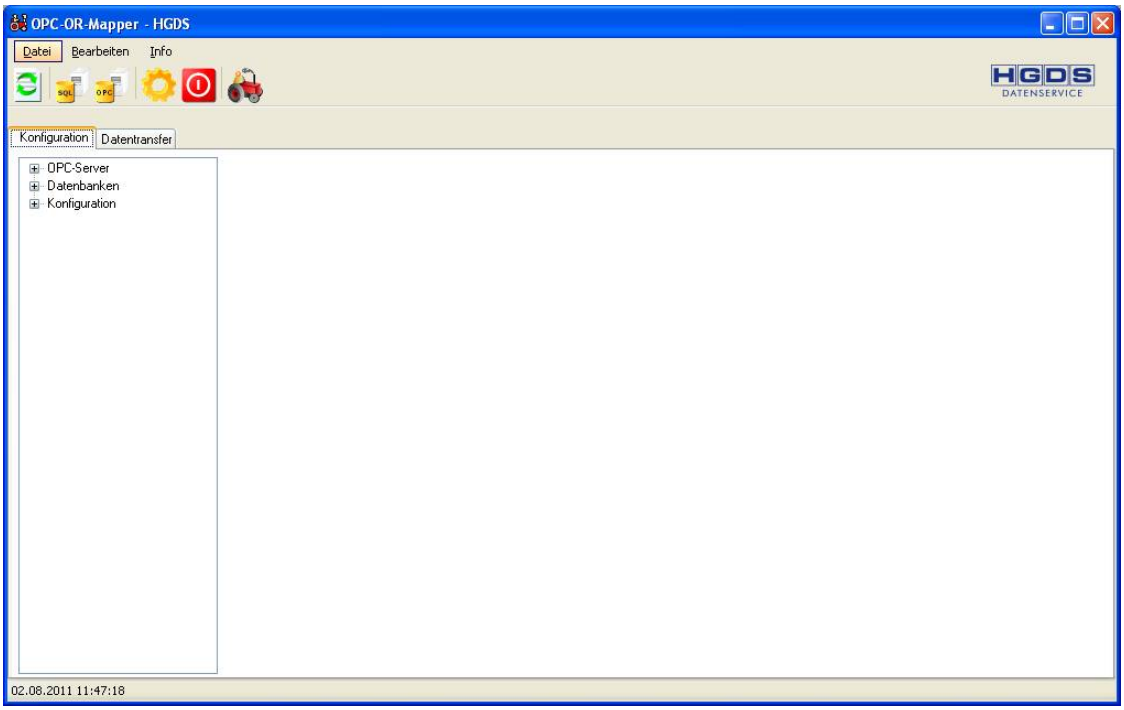

Abbildung A.7: Die Oberflächenstudie der Konfiguration

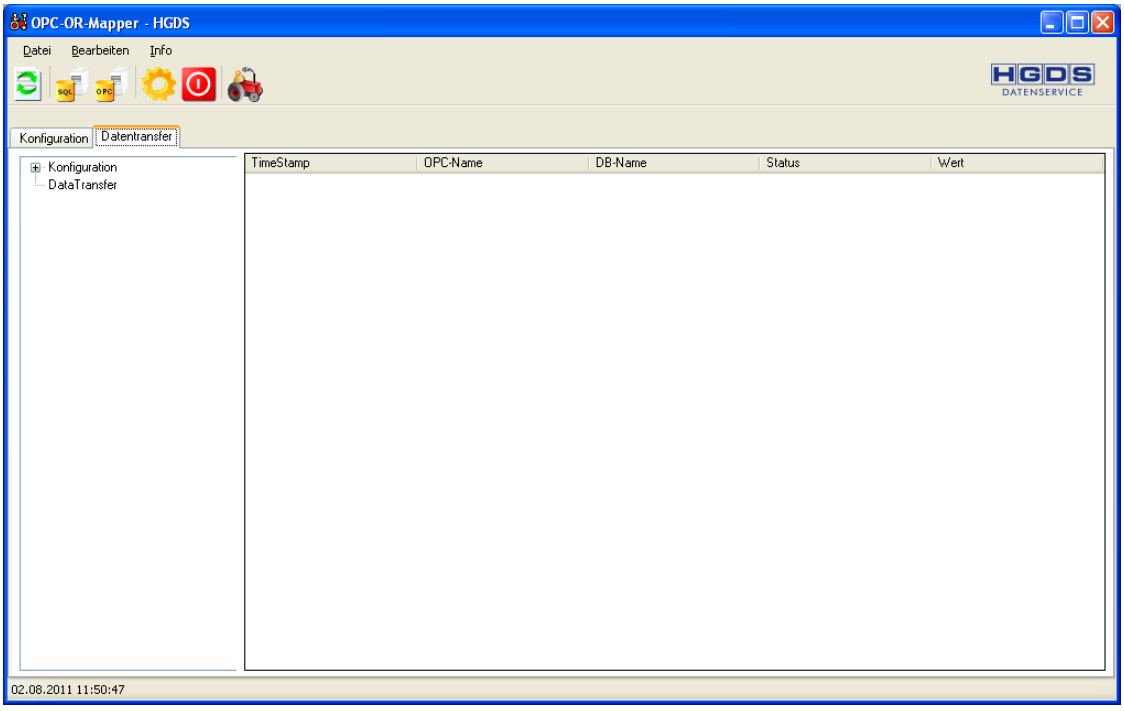

Abbildung A.8: Die Oberflächenstudie des Datentransfers

Abbildung A.9: Der Datenbankdialog

Abbildung A.10: Der OPC-Server-Dialog

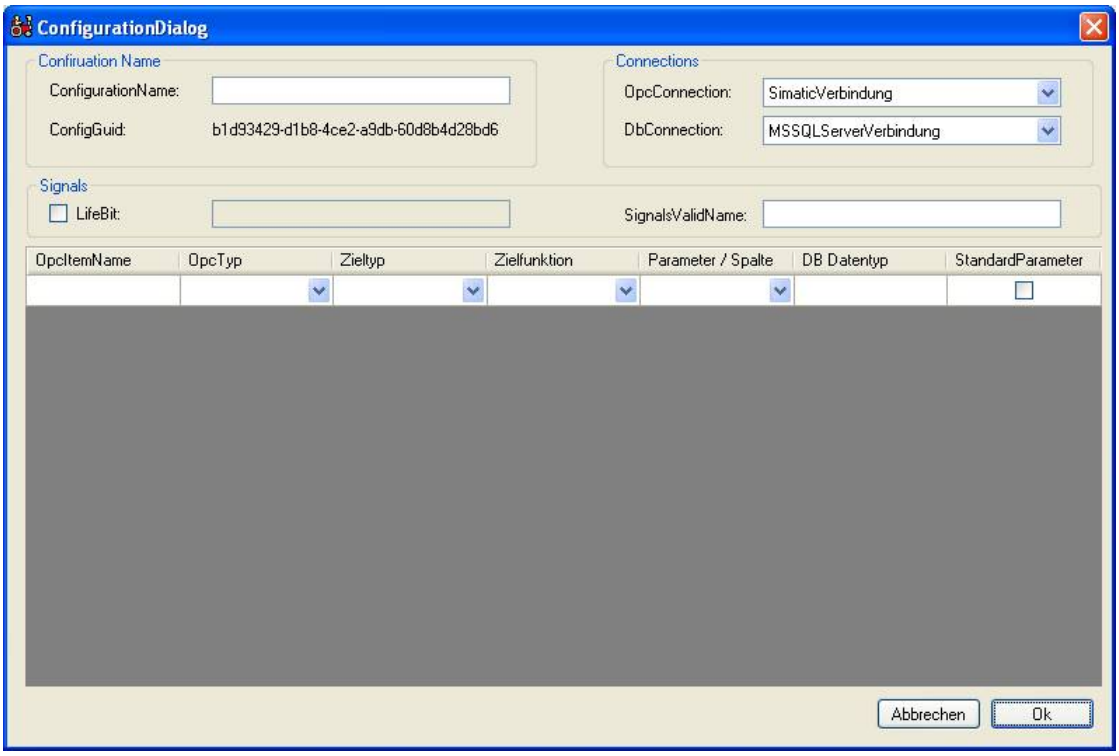

Abbildung A.11: Der Konfigurationsdialog

A.9 Nichtfunktionale Anforderungen

Die Anwendung verfügt über eine leicht verständliche Programmoberfläche. Um die Erweiterbarkeit zu gewährleisten, ist die Anwendung modular zu gestalten.

A.10 Technische Produktumgebung

A.10.1 Software

- Betriebssystem: Windwos 2000 oder höher
- .NET Framework 2.0

A.10.2 Hardware

- Prozessor 400Mhz oder höher
- RAM 128 MB oder höher
- Speicherplatz 100 MB

A.10.3 Orgware

Eine Verbindung zu einem Datenbankmanagementsystem muss vorhanden sein.

A.10.4 Produktschnittstellen

Die Benutzer die für den Zugriff auf das Datenbankmanagementsystem vorgesehen sind, müssen über alle notwendigen Rechte verfügen um alle Transaktion durchführen zu dürfen.

A.11 Spezielle Anforderungen an die Entwicklungsumgebung

Alle zur Entwicklung nötigen Programme sollten der Firma HGDS zur Verfügung stehen.

A.11.1 Software

- Visual Studio 2005
- .NET Framework 2.0

A.11.2 Hardware

Siehe A.10.2

A.11.3 Orgware

Siehe A.10.3

A.11.4 Entwicklungs-Schnittstellen

Es wird sich zweier Schnittstellen bedient:

- ein OPC-Client, der in der Lage ist, auf die Items vom OPC-Server zuzugreifen.
- eine Datenbankschnittstelle, die es ermöglicht, mit der Datenbank zu kommunizieren.

A.12 Gliederung in Teilprodukte

Das Produkt gliedert sich in die folgenden beiden Teilprodukte:

A.12.1 Framework

Das Framework ist der Kern der Anwendung, es stellt alle Funktionen zur Konfiguration und zum Datenaustausch zur Verfügung.

Die Konfiguration

Vor dem Datenaustausch steht die Konfiguration dessen. Dazu gehören die Konfiguration der Quelle und des Ziels, die Auswahl der Quell-Items und deren dazugehörige Zielfunktion bzw. Zieltabelle.

Der Datentransfer

Unter Verwendung der angelegten Daten werden die Daten aus der Automatisierungsanlage gelesen, verarbeitet und schließlich in die Datenbank geschrieben.

A.12.2 Programmoberfläche

Die Programmoberfläche vereinigt die beiden grundlegenden Aufgaben. Wichtig bei dieser Teilaufgabe ist es, dem Benutzer, die Konfiguration sowie auch die Verwaltung des Datenaustauschs ohne jeglichen Programmieraufwand zu ermöglichen.

Anhang B: Der Grobentwurf

Version	Stand	Änderung/ Ergänzung	Status
0.5	02.06.2011	Grobentwurf	abgeschlossen

Tabelle B.1: Grobentwurf Revisionstabelle

B.1 Einflussfaktoren

Die Einsatz-, Umgebungs-, und Randbedingungen wurden bereits im Pflichtenheft festgehalten. Das nachfolgende versteht sich als Zusammenfassung und Erweiterung des Pflichtenhefts. Insbesondere werden in den Use-Case-Beschreibungen die Geschäftsprozesse detailliert geschildert, zum Grobentwurf gehören weiterhin das Packagediagramm und ein erstes Sequenzdiagramm. Auf die Sequenzdiagramme wird im Feinentwurf näher eingegangen.

B.2 UseCase Beschreibungen

B.2.1 Verbindung zur Datenbankverwalten

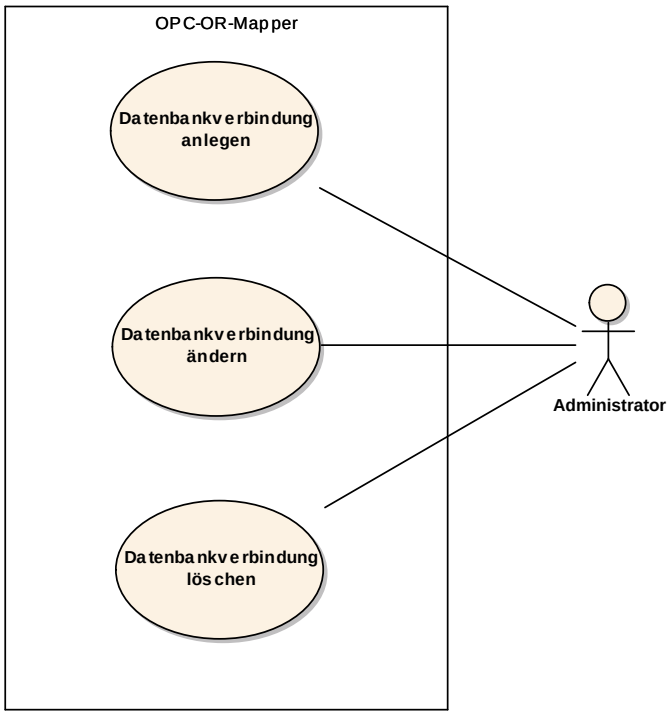

Abbildung B.1: Verbindung zur Datenbank verwalten

Pos	Name	Beschreibung
1	Geschäftsprozess:	Verbindung zur Datenbank herstellen
	Ziel:	Es wird eine Verbindung zu einer existierenden Datenbank hergestellt.
	Kategorie:	Sekundär
	Vorbedingung:	Keine
	Nachbedingung Erfolg:	Eine neue Verbindung zu einer Datenbank wurde in Form einer gespeicherten XML-Struktur angelegt.
	Nachbedingung Fehlschlag:	Es konnte keine neue Verbindung angelegt werden.
	Akteure:	Administrator
	Auslösendes Ereignis:	Es muss ein neues Datenziel angelegt werden, z.B. weil eine neue Konfiguration erstellt werden soll.
	Beschreibung:	• Dialog wird geöffnet, optional lässt sich diese Funktion auch über die Kommandozeile aufrufen. • Eingabe der unter Punkt /PD010/ benannten Daten • Übergabe der erfassten Daten und damit auslösen des Speichervorgangs via XML-File.
	Erweiterung	Nach dem Bestätigen der Verbindung wird durch Aufforderung ein Verbindungstest ausgeführt.
	Alternativen	keine

Tabelle B.2: UseCase-Beschreibung Datenbankverbindung herstellen

Pos	Name	Beschreibung
1	Geschäftsprozess:	Verbindung zur Datenbank ändern
	Ziel:	Es soll eine existierende Datenbank-Verbindung geändert werden.
	Kategorie:	Sekundär
	Vorbedingung:	Eine Datenbankverbindung existiert.
	Nachbedingung Erfolg:	Eine bestehende Datenbank-Verbindung wurde geändert.
	Nachbedingung Fehlschlag:	Die Verbindung konnte nicht verändert werden, alle Änderungen werden Rückgängig gemacht.
	Akteure:	Administrator
	Auslösendes Ereignis:	Eine bestehende Datenbank-Verbindung soll geändert werden, z.B. weil die Datenbank auf einem neuen Server umgezogen ist.
	Beschreibung:	• Auswahl der zu ändernden Datenbank-Verbindung • Nach dem Rechtsklick mit der Maus auf diese den Untermenüeintrag „Bearbeiten“ auswählen. • Dialog wird geöffnet • Die Daten der Datenbank-Verbindung werden in den dafür vorgesehenen Feldern angezeigt. • Die zu ändernden Daten werden überschrieben. • Bestätigen der Änderung.
	Erweiterung	Nach dem Bestätigen der Verbindung wird durch Aufforderung ein Verbindungstest ausgeführt.
	Alternativen	Die zu ändernde Datenbankverbindung wird gelöscht, anschließend wird eine neue Datenbankverbindung angelegt.

Tabelle B.3: UseCase-Beschreibung Datenbankverbindung ändern

Pos	Name	Beschreibung
1	Geschäftsprozess:	Verbindung zur Datenbank löschen
	Ziel:	Es soll eine existierende Datenbank-Verbindung gelöscht werden.
	Kategorie:	Sekundär
	Vorbedingung:	Eine Datenbankverbindung existiert.
	Nachbedingung Erfolg:	Eine bestehende Datenbank-Verbindung wurde gelöscht.
	Nachbedingung Fehlschlag:	Die Verbindung konnte nicht gelöscht werden, die Verbindung besteht weiterhin.
	Akteure:	Administrator
	Auslösendes Ereignis:	Eine bestehende Datenbank-Verbindung soll gelöscht werden, z.B. weil die Datenbank nicht mehr existiert.
	Beschreibung:	• Auswahl der zu löschenden Datenbank-Verbindung • Nach dem Rechtsklick mit der Maus auf diese den Untermenüeintrag „Löschen“ auswählen. • Bestätigung des Löschvorgangs. • Die Verbindung und die dazugehörige XML-Struktur wurde gelöscht.
	Erweiterung	Sollte nach dem Löschen keine weitere Datenbank-Verbindung mehr existieren, wird auf Nachfrage der Dialog geöffnet, um eine neue Verbindung anzulegen.
	Alternativen	keine

Tabelle B.4: UseCase-Beschreibung Datenbankverbindung löschen

B.2.2 Verbindung zum OPC-Server verwalten

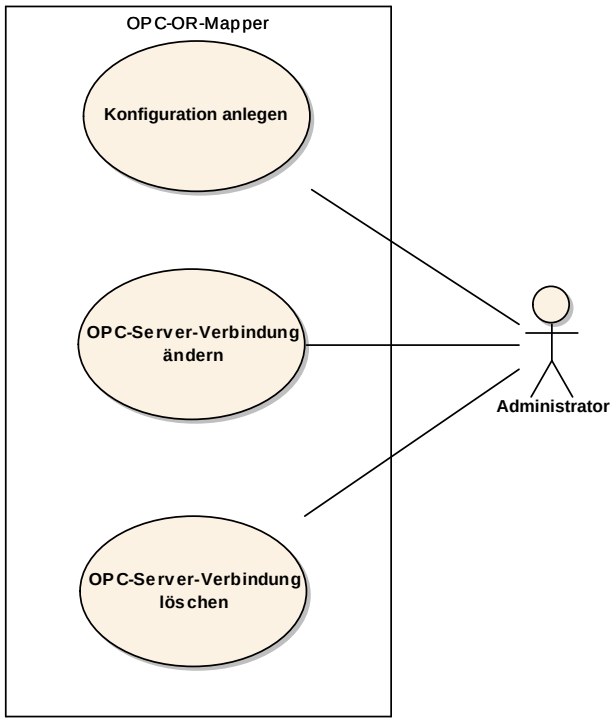

Abbildung B.2: Verbindung zum OPC-Server verwalten

Pos	Name	Beschreibung
1	Geschäftsprozess:	Verbindung zum OPC-Server herstellen
	Ziel:	Es wird eine Verbindung zu einem existierenden OPC-Server hergestellt.
	Kategorie:	Sekundär
	Vorbedingung:	Keine
	Nachbedingung Erfolg:	Eine neue Verbindung zu einem OPC-Server wurde angelegt.
	Nachbedingung Fehlschlag:	Es konnte keine neue Verbindung angelegt werden.
	Akteure:	Administrator
	Auslösendes Ereignis:	Es muss eine neue Datenquelle angelegt werden z.B. weil diese für eine neue Konfiguration benötigt werden.
	Beschreibung:	• Dialog wird geöffnet, optional lässt sich diese Funktion auch über die Kommandozeile aufrufen. • Eingabe der unter Punkt /PD020/ benannten Daten • Übergabe der erfassten Daten und damit auslösen des Speichervorgangs via XML-File.
	Erweiterung	Nach dem Bestätigen der Verbindung wird durch Aufforderung ein Verbindungstest ausgeführt.
	Alternativen	keine

Tabelle B.5: UseCase-Beschreibung Verbindung zum OPC-Server herstellen

Pos	Name	Beschreibung
1	Geschäftsprozess:	Verbindung zum OPC-Server ändern
	Ziel:	Es soll eine existierende OPC-Server-Verbindung geändert werden.
	Kategorie:	Sekundär
	Vorbedingung:	Eine OPC-Server-Verbindung existiert.
	Nachbedingung Erfolg:	Eine bestehende OPC-Server-Verbindung wurde geändert.
	Nachbedingung Fehlschlag:	Die Verbindung konnte nicht verändert werden, alle Änderungen werden Rückgängig gemacht.
	Akteure:	Administrator
	Auslösendes Ereignis:	Eine bestehende OPC-Server-Verbindung soll geändert werden, z.B. weil der OPC-Server auf einen neuen Server umgezogen ist.
	Beschreibung:	• Auswahl der zu ändernden Verbindung • Nach dem Rechtsklick mit der Maus auf diese den Untermenüeintrag „Bearbeiten“ auswählen. • Dialog wird geöffnet • Die Daten der OPC-Server-Verbindung werden in den dafür vorgesehenen Feldern angezeigt. • Die zu ändernden Daten werden überschrieben. • Bestätigen der Änderung
	Erweiterung	Nach dem Bestätigen der Verbindung wird durch Aufforderung ein Verbindungstest ausgeführt.
	Alternativen	Die zu ändernde OPC-Server-Verbindung wird gelöscht, anschließend wird eine neue OPC-Server-Verbindung angelegt.

Tabelle B.6: UseCase-Beschreibung Verbindung zum OPC-Server ändern

Pos	Name	Beschreibung
1	Geschäftsprozess:	Verbindung zum OPC-Server löschen
	Ziel:	Es soll eine existierende OPC-Server-Verbindung gelöscht werden.
	Kategorie:	Sekundär
	Vorbedingung:	Eine OPC-Server-Verbindung muss existieren.
	Nachbedingung Erfolg:	Eine bestehende OPC-Server-Verbindung wurde gelöscht.
	Nachbedingung Fehlschlag:	Die Verbindung konnte nicht gelöscht werden, die Verbindung existiert damit weiterhin.
	Akteure:	Administrator
	Auslösendes Ereignis:	Eine bestehende OPC-Server-Verbindung soll gelöscht werden, z.B. weil der OPC-Server nicht mehr existiert.
	Beschreibung:	• Auswahl der zu löschenden OPC-Server-Verbindung • Nach dem Rechtsklick mit der Maus auf diese den Untermenüeintrag „Löschen“ auswählen. • Dialog wird geöffnet mit welchem das Löschen bestätigt wird • Mit dem Bestätigen des Löschvorgangs wurde die Verbindung und das XML-Konstrukt gelöscht.
	Erweiterung	Sollte keine Verbindung mehr bestehen, wird auf Nachfrage eine neue Verbindung angelegt.
	Alternativen	Keine.

Tabelle B.7: UseCase-Beschreibung Verbindung zum OPC-Server löschen

B.2.3 Konfiguration verwalten

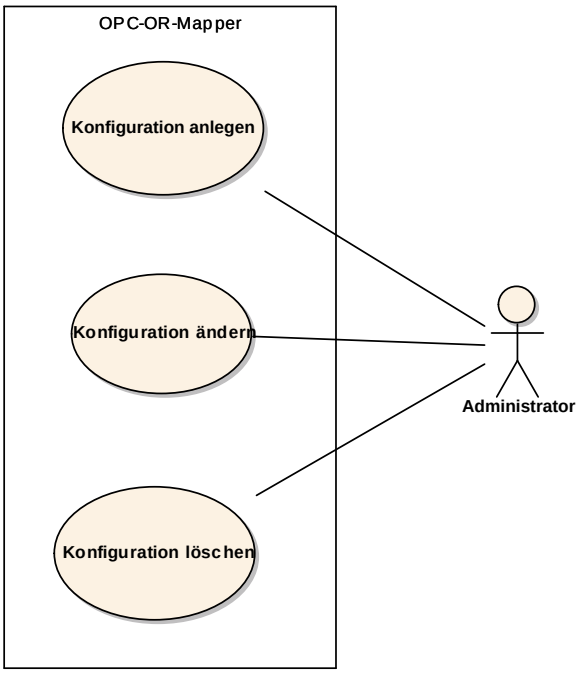

Abbildung B.3: Konfiguration verwalten

Pos	Name	Beschreibung
1	Geschäftsprozess:	Konfiguration erstellen
	Ziel:	Eine Konfiguration wird erstellt, die die spätere Grundlage für den Datenaustausch bildet. In dieser Konfiguration werden die Verbindung zum OPC-Server, die Verbindung zur Datenbank, die Auswahl der Daten sowie deren Zuordnung festgehalten. Abschließend wird diese erstellte Konfiguration in einer XMLStruktur gespeichert.
	Kategorie:	Primär
	Vorbedingung:	Eine Verbindung zum OPC-Server existiert. Eine Verbindung zur Datenbank existiert.
	Nachbedingung Erfolg:	Eine neue Konfiguration wurde erstellt.
	Nachbedingung Fehlschlag:	Die neue Konfiguration konnte nicht erstellt werden.
	Akteure:	Administrator
	Auslösendes Ereignis:	Für einen neuen Datentransfer wird eine neue Konfiguration benötigt. Für diesen sind zunächst die Parameter zu wählen und festzulegen.
	Beschreibung:	• Auswahl der OPC-Server-Verbindung • Auswahl der Datenbank-Verbindung • Selektieren der Daten vom OPC-Server • Selektieren der Stored Procedures bzw. Tabellen der ausgewählten Datenbankverbindung. • Zuordnung der ausgewählten Daten des OPC-Servers mit den zuvor ausgewählten Datenbank-Objekten des Ziels. • Die Konfiguration wird angelegt. • Nach dem Bestätigen der Konfiguration wird diese in einer XML-Struktur gespeichert.
	Erweiterung	Keine
	Alternativen	Keine.

Tabelle B.8: UseCase-Beschreibung Konfiguration erstellen

Pos	Name	Beschreibung
1	Geschäftsprozess:	Konfiguration ändern
	Ziel:	Es soll eine existierende Konfiguration geändert werden.
	Kategorie:	Primär
	Vorbedingung:	Eine Konfiguration existiert. Eine OPC-Server-Verbindung existiert. Eine Datenbank-Verbindung existiert.
	Nachbedingung Erfolg:	Eine bestehende Konfiguration wurde geändert.
	Nachbedingung Fehlschlag:	Die Konfiguration konnte nicht verändert werden, alle Änderungen werden Rückgängig gemacht.
	Akteure:	Administrator
	Auslösendes Ereignis:	Eine bestehende Konfiguration soll geändert werden, z.B. weil sich die zu überwachenden Quelldaten geändert haben.
	Beschreibung:	• Auswahl der zu ändernden Konfiguration • Nach dem Rechtsklick mit der Maus auf diese den Untermenüeintrag „Bearbeiten“ auswählen. • Dialog wird geöffnet • Die Daten der Konfiguration werden in den dafür vorgesehenen Feldern angezeigt. • Die zu ändernden Daten werden überschrieben. • Die zu ändernden Daten werden überschrieben.
	Erweiterung	Keine
	Alternativen	Die zu ändernde Konfiguration wird gelöscht, anschließend wird eine neue Konfiguration angelegt.

Tabelle B.9: UseCase-Beschreibung Konfiguration ändern

Pos	Name	Beschreibung
1	Geschäftsprozess:	Konfiguration löschen
	Ziel:	Es soll eine existierende Konfiguration gelöscht werden.
	Kategorie:	Primär
	Vorbedingung:	Eine Konfiguration muss existieren
	Nachbedingung Erfolg:	Eine bestehende Konfiguration wurde gelöscht.
	Nachbedingung Fehlschlag:	Die Konfiguration konnte nicht gelöscht werden, und besteht weiterhin.
	Akteure:	Administrator
	Auslösendes Ereignis:	Eine Konfiguration soll gelöscht werden, z.B. weil sie nicht mehr benötigt wird.
	Beschreibung:	• Auswahl der zu löschenden Konfiguration • Nach dem Rechtsklick mit der Maus auf diese den Untermenüeintrag „Löschen“ auswählen. • Nach dem Bestätigen des Löschvorgangs werden die Konfiguration und das dazugehörige XML-Konstrukt gespeichert.
	Erweiterung	Keine
	Alternativen	Keine

Tabelle B.10: UseCase-Beschreibung Verbindung zum OPC-Server löschen

B.2.4 Verwaltung des Datentranfers

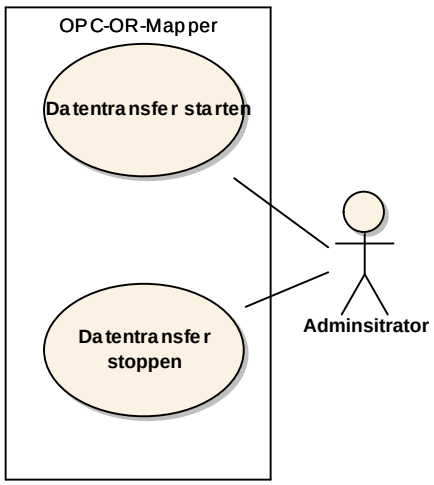

Abbildung B.4: Verwaltung des Datentransfers

Pos	Name	Beschreibung
1	Geschäftsprozess:	Datentransfer starten
	Ziel:	Ein Datentransfer wird gestartet.
	Kategorie:	Primär
	Vorbedingung:	Eine Konfiguration ist vorhanden.
	Nachbedingung Erfolg:	Der beginnt mit dem zyklischen Lesen und Schreiben der konfigurierten Daten.
	Nachbedingung Fehlschlag:	Der Datentransfer konnte nicht gestartet werden.
	Akteure:	Administrator
	Auslösendes Ereignis:	Ein Datentransfer soll gestartet werden z.B. weil die Daten des OPC-Servers in der Datenbank benötigt werden.
	Beschreibung:	• Auswahl der zu startenden Konfiguration • Nach dem Rechtsklick mit der Maus auf diese den Untermenüeintrag „Starten“ auswählen. • Nach dem Bestätigen des Löschvorgangs werden die Konfiguration und das dazugehörige XML-Konstrukt gespeichert.
	Erweiterung	Keine
	Alternativen	Keine

Tabelle B.11: UseCase-Beschreibung Datentransfer starten

Pos	Name	Beschreibung
1	Geschäftsprozess:	Datentransfer stoppen
	Ziel:	Ein laufender Datentransfer wird mit dieser Funktion gestoppt.
	Kategorie:	Primär
	Vorbedingung:	Ein aktiver Datentransfer existiert.
	Nachbedingung Erfolg:	Ein aktiver Datentransfer wird gestoppt.
	Nachbedingung Fehlschlag:	Der Datentransfer ist weiterhin aktiv.
	Akteure:	Administrator
	Auslösendes Ereignis:	Ein aktiver Datentransfer soll gestoppt werden, z.B. weil an der Konfiguration Änderungen vorgenommen werden sollen.
	Beschreibung:	• Auswahl des zu stoppenden Datentransfers • Nach dem Rechtsklick mit der Maus auf diesen den Untermenüeintrag „Stoppen“ auswählen. • Bestätigen des Vorgangs.
	Erweiterung	Keine
	Alternativen	Keine

Tabelle B.12: UseCase-Beschreibung Datentransfer stoppen

B.3 Packagediagramm

In dem folgenden Diagramm ist die Teilung der Benutzungsoberfläche vom Framework ersichtlich. Daraus geht hervor, dass auf der Seite der Anwendung nach dem MVC-Prinzip programmiert wird, während sich das Framework, dass alle Funktionen zum Datenaustausch und zur Konfiguration zur Verfügung stellt, auf die beiden Connector-Schnittstellen zugreift.

B.3.1 Diagramm

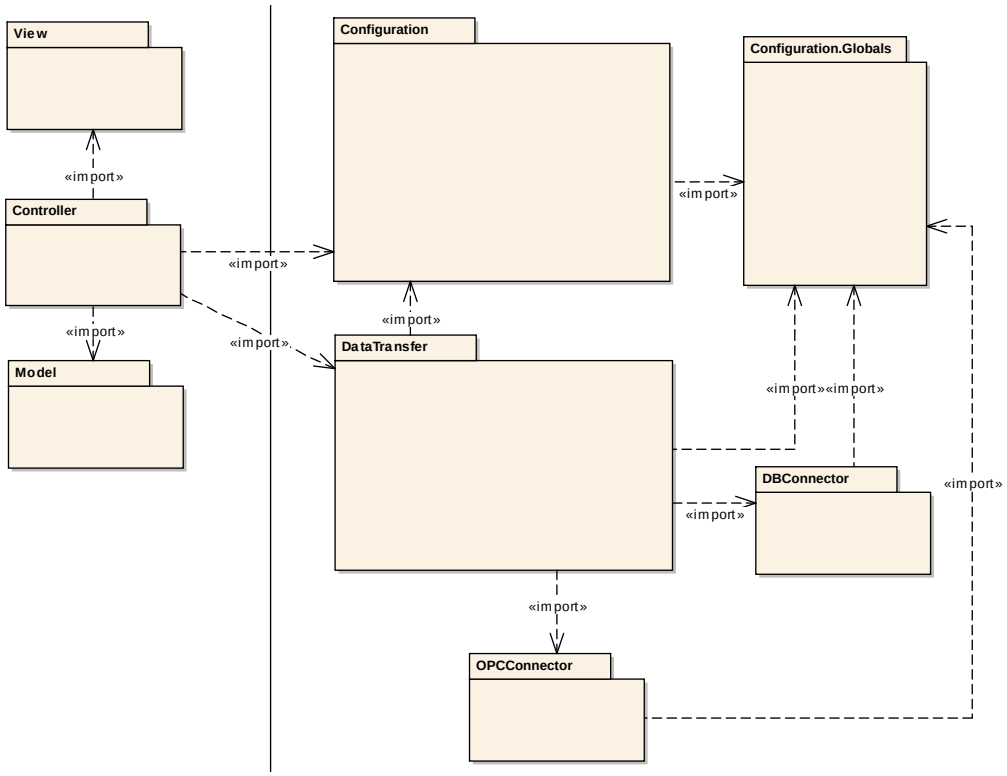

Abbildung B.5: Das PackageDiagramm

B.3.2 Anmerkungen

Pos	Package	Beschreibung
1	Allgemein:	Unabhängig von den Paketen ist vor dem Schreiben im Filesystem generell zu prüfen, ob die dafür notwendigen Rechte und der benötigte Speicherplatz zur Verfügung steht.
	DBConnector:	1. Wurde mit diesem Namen bereits eine Verbindung angelegt (Name muss eineindeutig sein)? 2. Im Falle eines gelöschten Verbindungsfiles muss dies durch eine Fehlerbehandlung abgefangen werden.
	OPCConnector:	1. Existiert bereits eine Verbindung mit diesem Namen? (Name muss eineindeutig sein). 2. Eine Fehlerbehandlung für den Fall eines gelöschten Verbindungsfiles muss vorgesehen werden.

Pos	Package	Beschreibung
	Framework:	1. Das Löschen, beispielsweise einer Konfiguration, muss das Löschen des dazugehörigen XML-Konstrukts nach sich ziehen, da ansonsten Namen nicht erneut vergeben werden können. 2. Bei der Zuordnung der OPC-Items zu den Datenbankobjekten ist darauf zu achten, dass die richtigen Typen verwendet werden z.B. keine Zeichenketten in Zahlenfelder zu schreiben. Eventuell ist das nicht vorhandensein, solcher Typen durch Typkonvertierungen möglich. 3. Auch das Framework muss die Fehlerbehandlungdurchführen, für den Fall eines gelöschten Configfiles.
	Modell	Das Model gehört zur Anwendungsseite der Schnittstelle und ist damit als flüchtig anzusehen. Die Datenhaltung in Form der gespeicherten XML-Strukturen übernimmt die Schnittstelle.
	View	In dieser Ausbaustufe ist lediglich die Bedienung via GUI vorgesehen, dennoch gilt es, die Möglichkeiten der Bedienung mit Hilfe einer Konsole nicht außer Acht zu lassen.
	Controller	Wie auch in der View muss auch bei der Gestaltung des Controllers darauf geachtet werden, dass in späteren Entwicklungsstufen der Einsatz einer Konsole als Frontend angedacht ist. Hierbei gilt es zunächst zu prüfen wie das Programm gestartet wurde. Zum einen per Konsole (also mit Parametern) oder via GUI(ohne Parameter).

Tabelle B.13: Anmerkungen Packagediagramm

B.4 Klassendiagramm

Um die Übersicht des Klassendiagramms zu erhalten, wurde dies in die Teile OpcConnector, DbConnector, Framework und Anwendung aufgeteilt. Wobei das Paket Anwendung, wie der Name schon sagt, das Frontend für die Schnittstelle darstellt.

B.4.1 Connectoren

Das Paket DbConnector und dessen Klasse DbConnection nutzt die bereits vorhanden Bibliotheken „HGDS.DE.Data.Data.Access“ der Firma HGDS.

Das Paket OpcConnector und dessen Klasse DbConnection nutzt die bereits vorhanden Bibliotheken „HGDS.DE.OPC“ der Firma HGDS.

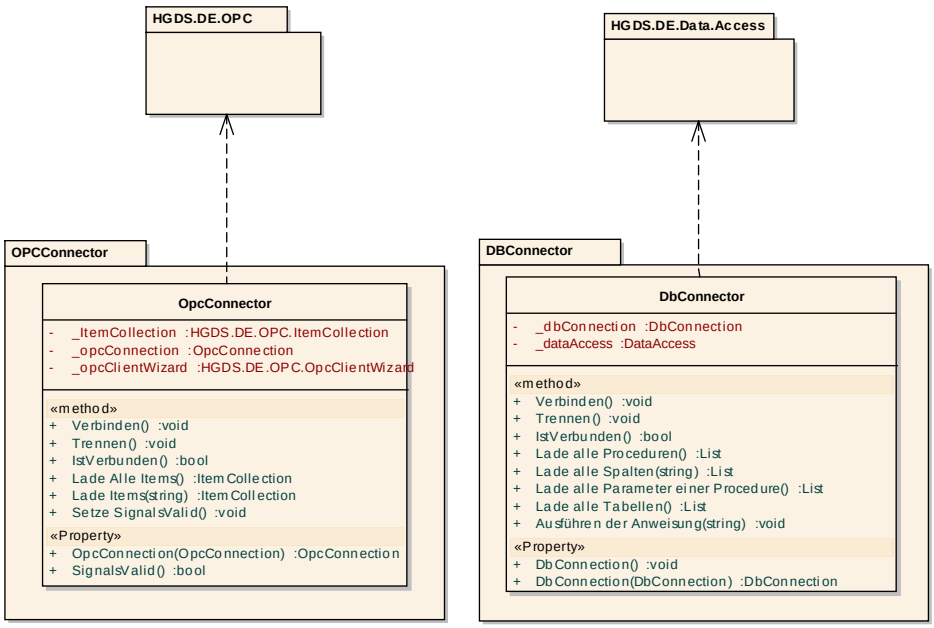

Abbildung B.6: Klassendiagramm DbConnector

B.4.2 Konfiguration

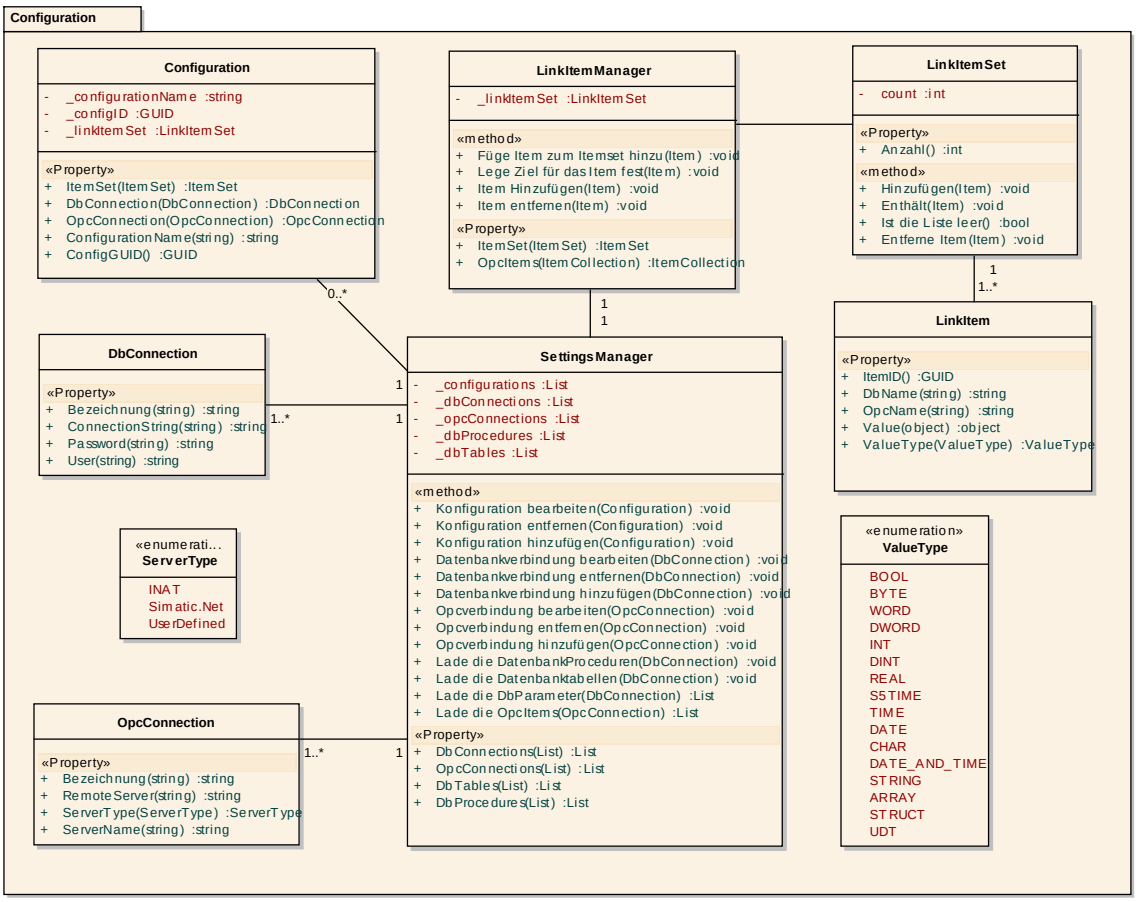

Abbildung B.7: Das Paket Configuration in der objektorientierten Analyse

B.4.3 Datentransfer

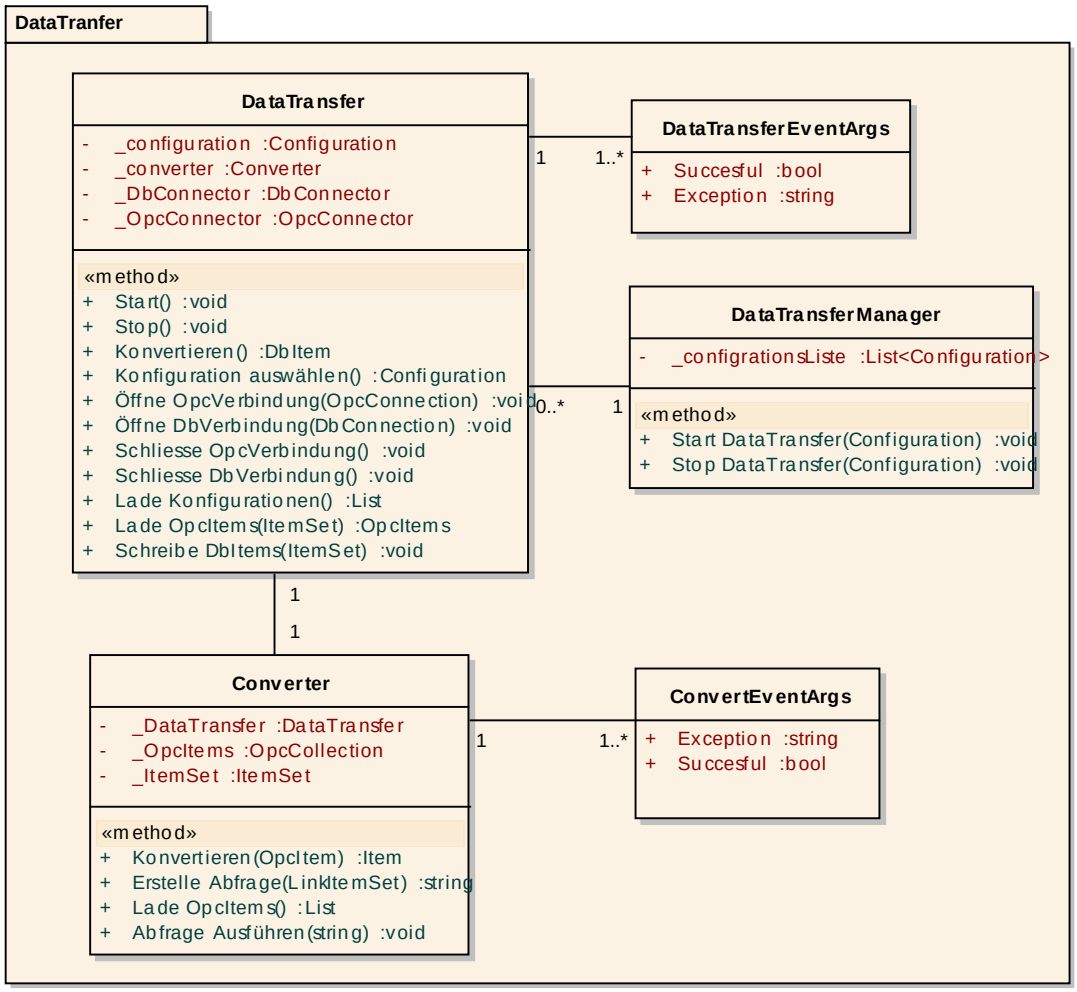

Abbildung B.8: Das Paket DataTransfer in der objektorientierten Analyse

B.4.4 Anwendung

Die Anwendung wird nach dem MVC-Prinzip konzipiert. Sie nutzt das Framework, das hier als „Black Box“ dargestellt wird.

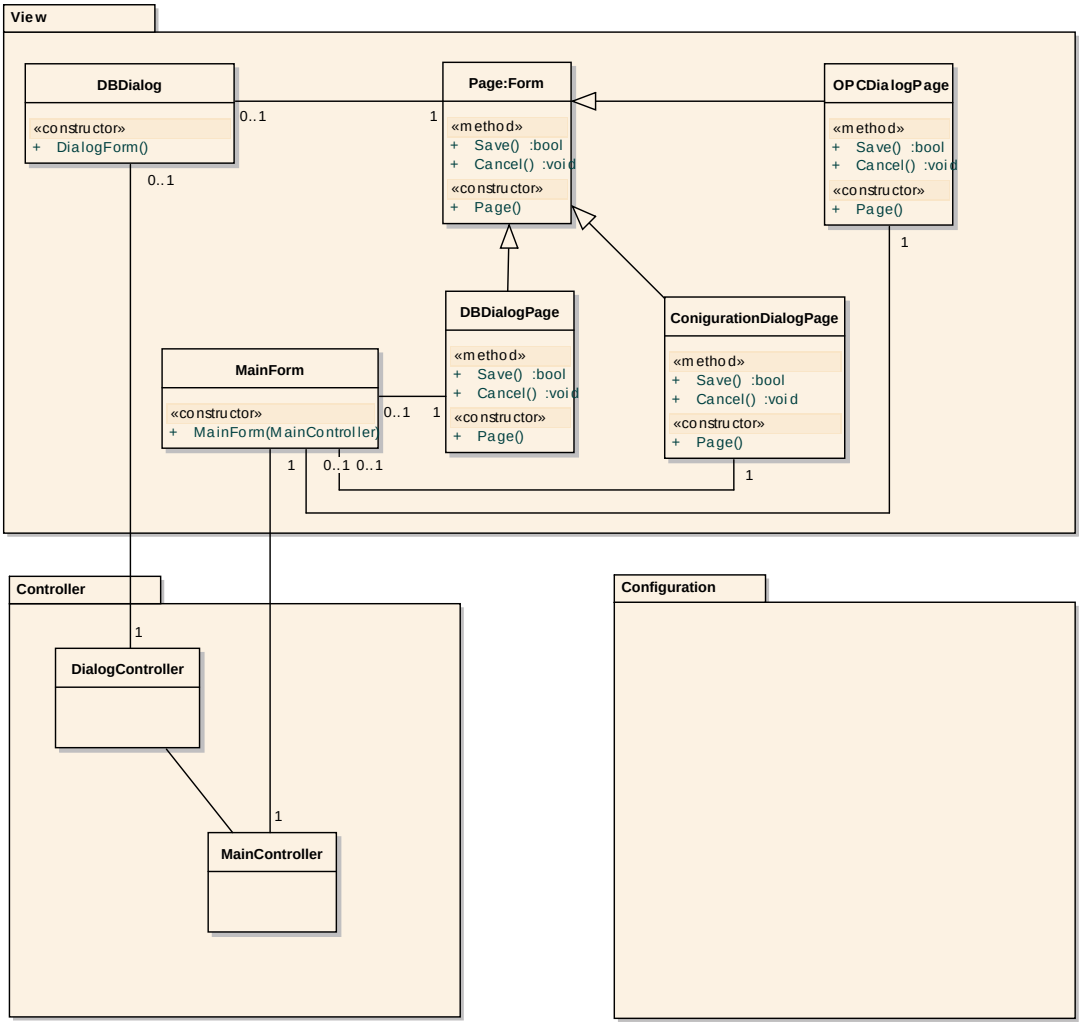

Abbildung B.9: Das Frontend in der objektorientierten Analyse

B.5 Sequenzdiagramm

Als erstes Sequenzdiagramm dient das Erstellen einer neuen Konfiguration. Dabei wird, wie im Diagramm ersichtlich, zunächst der Haupteinstieg für die Bibliothek OpcOrDataAccess instanziert. Von diesem Objekt aus lässt sich mit Hilfe einer Methode eine neue Konfiguration erstellen. Nachdem die Konfiguration angelegt wurde, müssen die OpcConnection und die DbConnection ausgewählt werden. Sollten diese noch nicht vorhanden sein, müssen sie angelegt werden. Nachdem die Quell- und die Zielverbindung festgelegt wurden, wird eine Liste mit dem OpcItems sowie ein Schema der Datenbank geladen. Diese müssen nun selektiert und verknüpft werden.

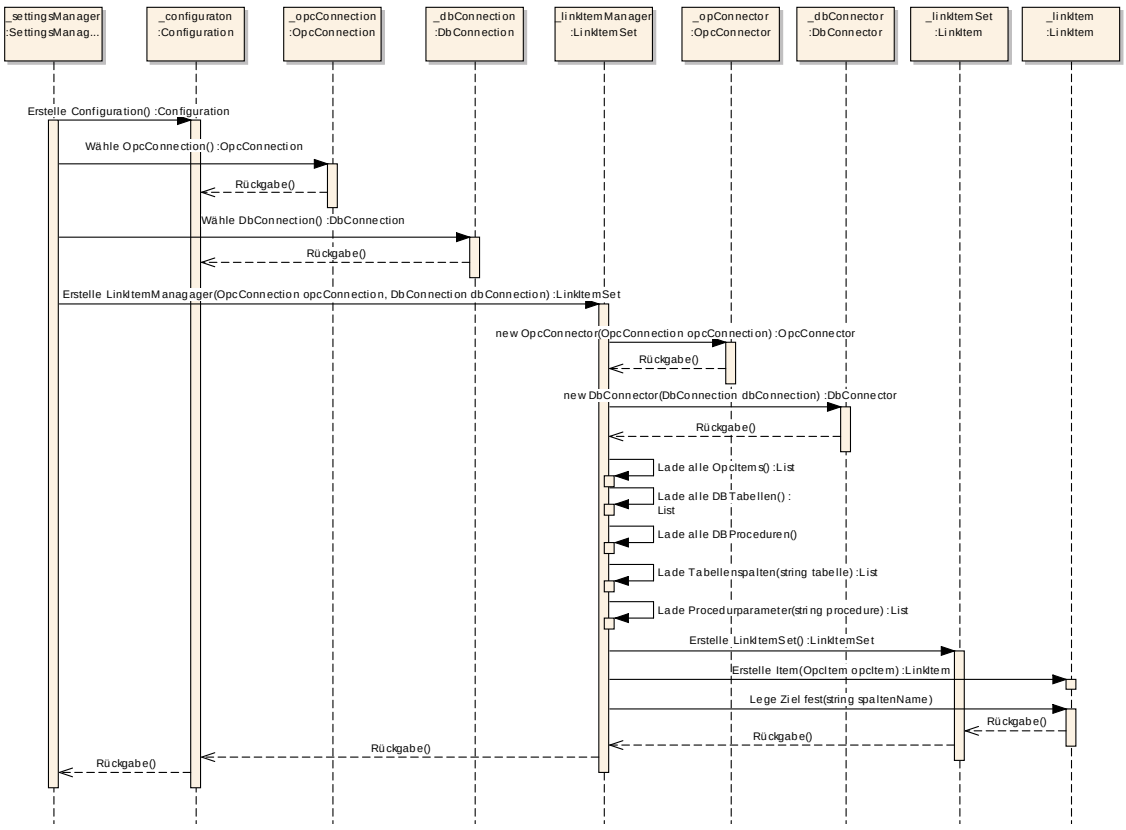

Abbildung B.10: Sequenzdiagramm Anlegen einer Konfiguration

Anhang C: Der Feinentwurf

Version	Stand	Änderung/ Ergänzung	Status
0.6	15.06.2011	Feinentwurf	vorläufig
0.7	01.07.2011	Feinentwurf	abgeschlossen

Tabelle C.1: Feinentwurf Revisionstabelle

C.1 Einleitung

Der Feinentwurf wird als der Abschluss der Objektorientierten Analyse und damit der Entwurfsphase angesehen. Bestandteil des Feinentwurfs sind das Klassendiagramm, die Aktivitätsdiagramme sowie die Sequenzdiagramme. Die Aktivitäts- und Sequenzdiagramme verstehen sich als Erläuterung des Klassendiagramms. Der Feinentwurf gliedert sich nach den Funktionen, dass bedeutet, zu jeder Funktion des Klassendiagramms gibt es unter einer eigenen Überschrift das dazugehörige Aktivitäts- und Sequenzdiagramm.

C.2 Die Klassendiagramme

Dieses Kapitel enthält die verfeinerten Klassendiagramme, die an die Programmiersprache angepasst wurden.[48]

[48] Alle weiteren Klassendiagramme befinden sich im Kapitel:5.4.2

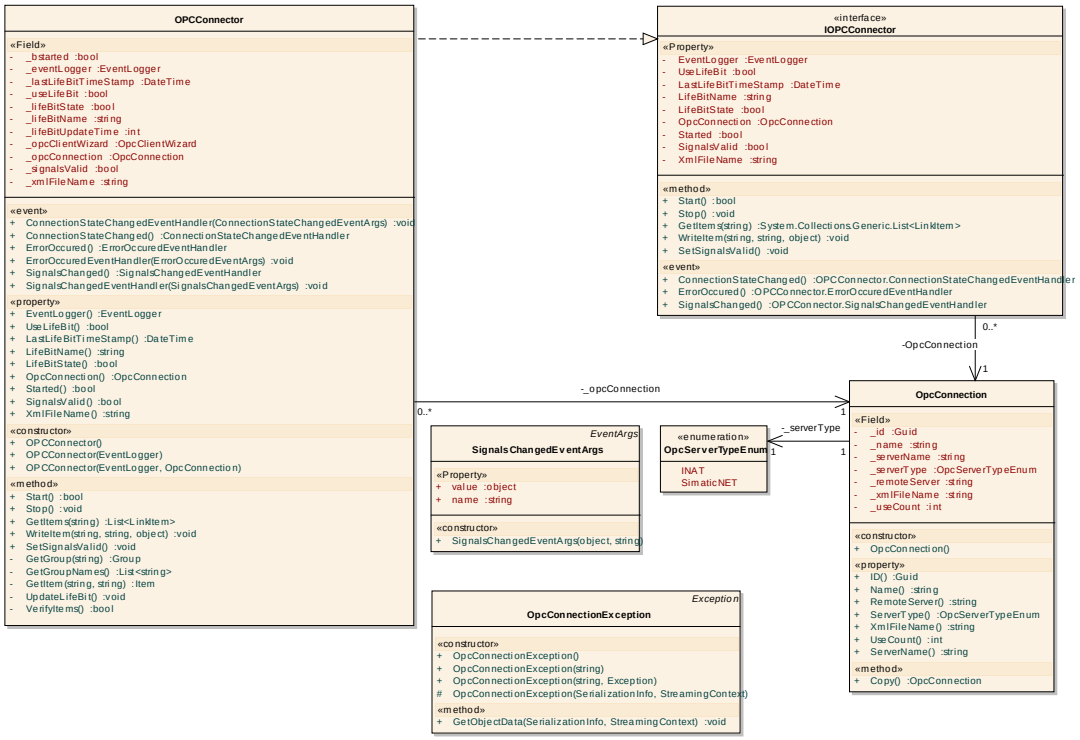

Abbildung C.1: Der OpcConnector im objektorientierten Design

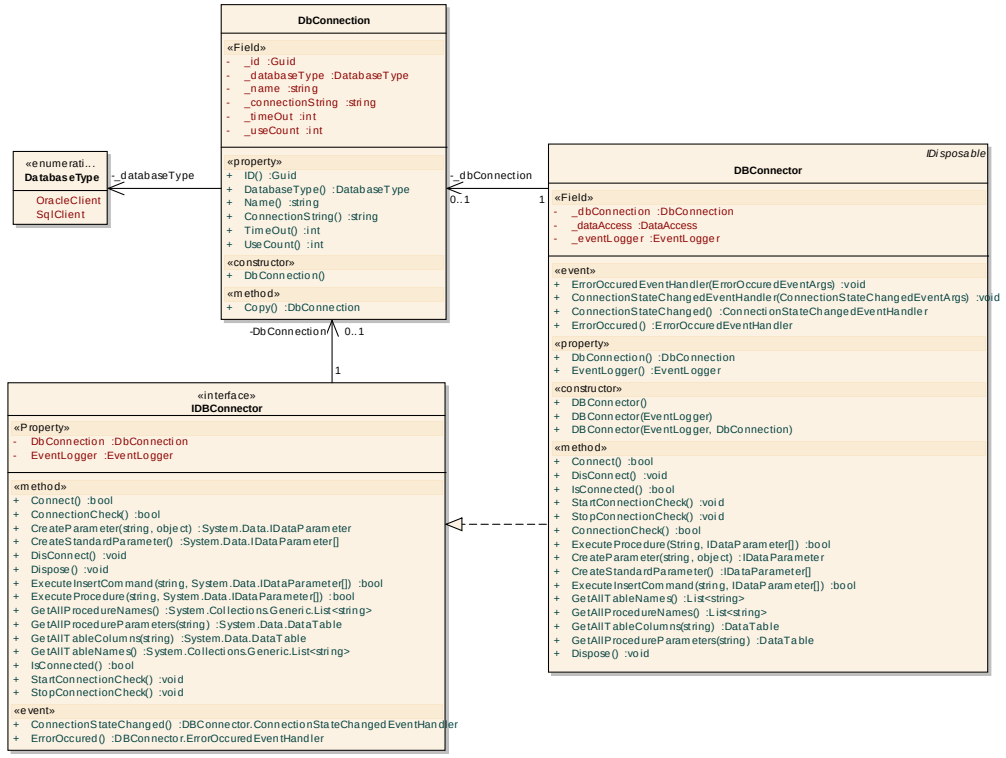

Abbildung C.2: Der DbConnector im objektorientierten Design

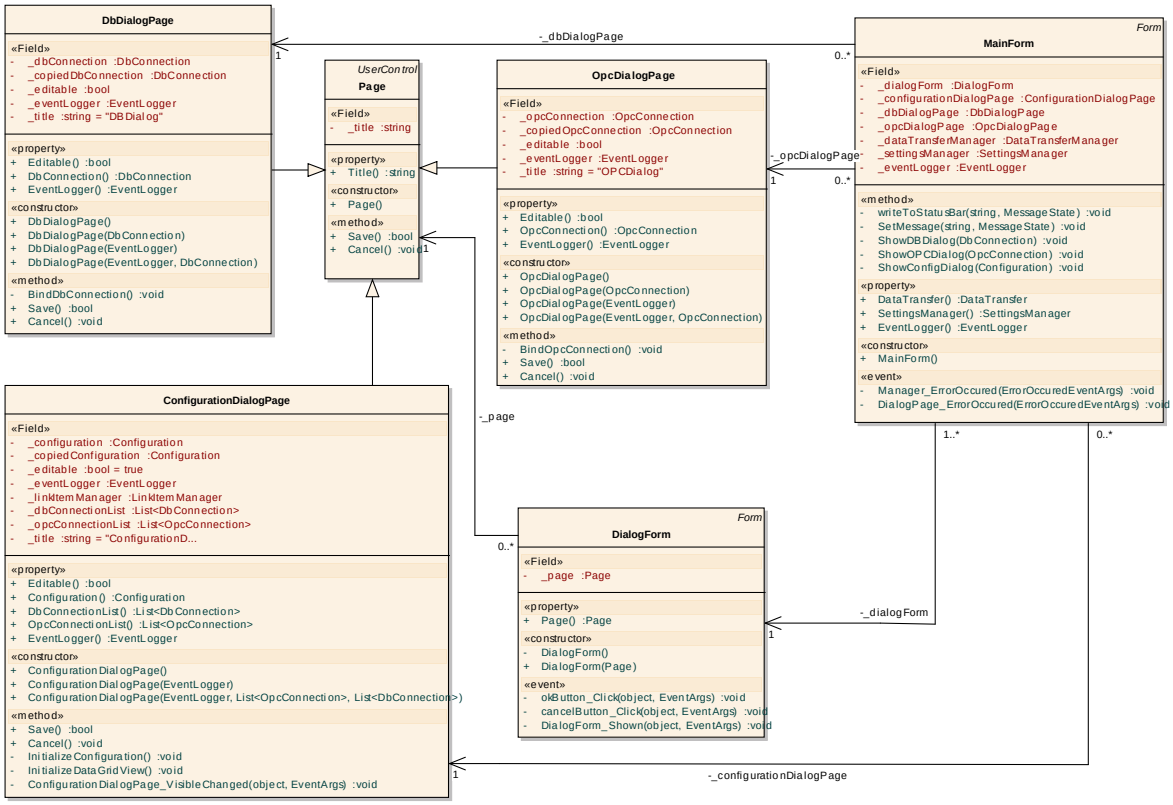

Abbildung C.3: Das GUI im objektorientierten Design

C.3 Die Aktivitätsdiagramme

In der UML werden Zustandsdiagramme verwendet, um asynchrone Ereignisse zu modellieren. Repräsentieren alle oder die meisten Zustände aber Schritte in der Ausführung eines Algorithmus oder eines Geschäftsprozesses, dann liegt ein Aktions-Zustand (action state) vor. Ein Diagrammm, bestehend aus Aktions-Zuständen und Zustandsübergängen, heißt in der UML **Aktivitätsdiagramm** (activity diagram) und stellt eine Variante des Zustandsautomaten dar. Aktivitätsdiagramme sind nicht einzelnen Klassen zugeordnet.[49]

[49] [HeBa00] Seite 334/335

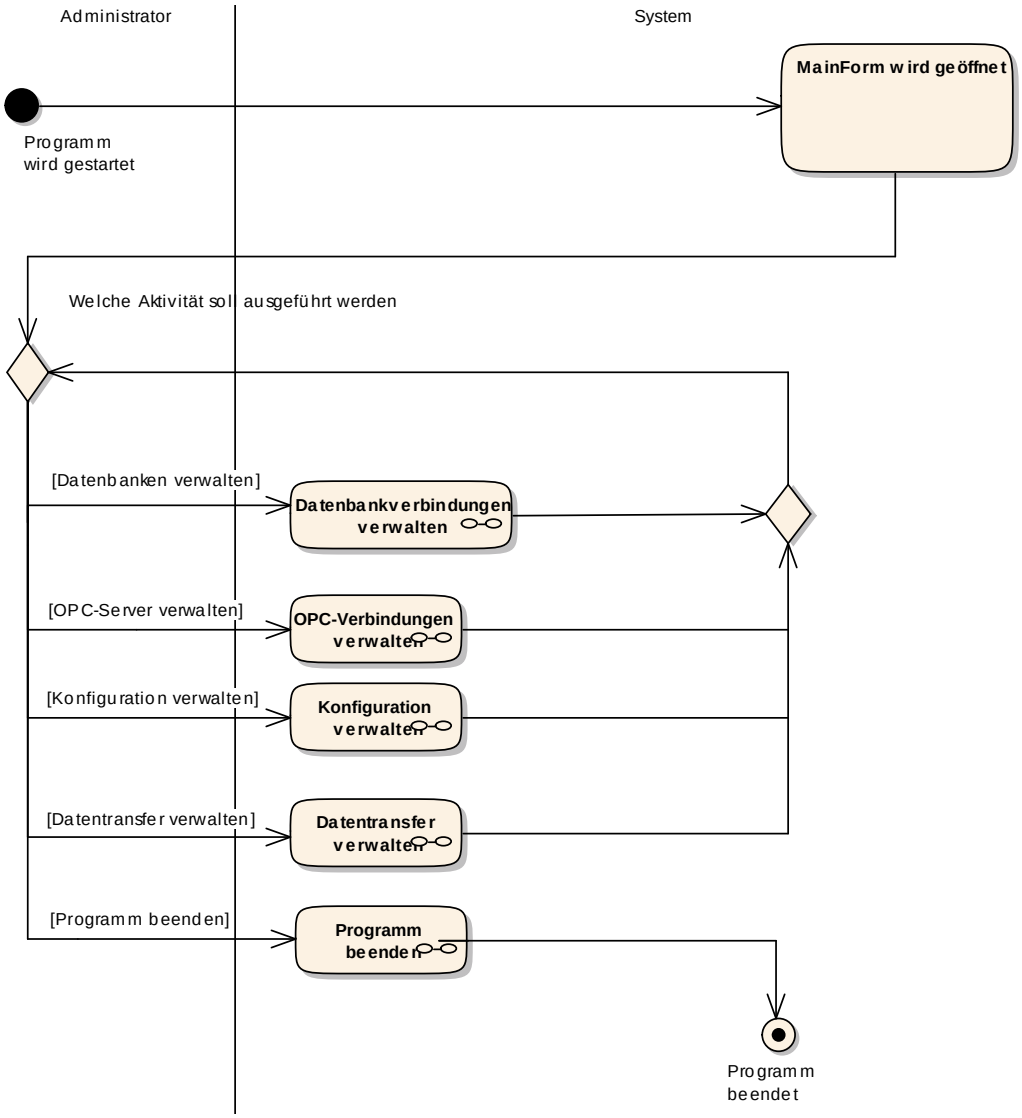

Abbildung C.4: Das Aktivitätsdiagramm für die Gesamtübersicht

C.3.1 Datenbankverbindung verwalten

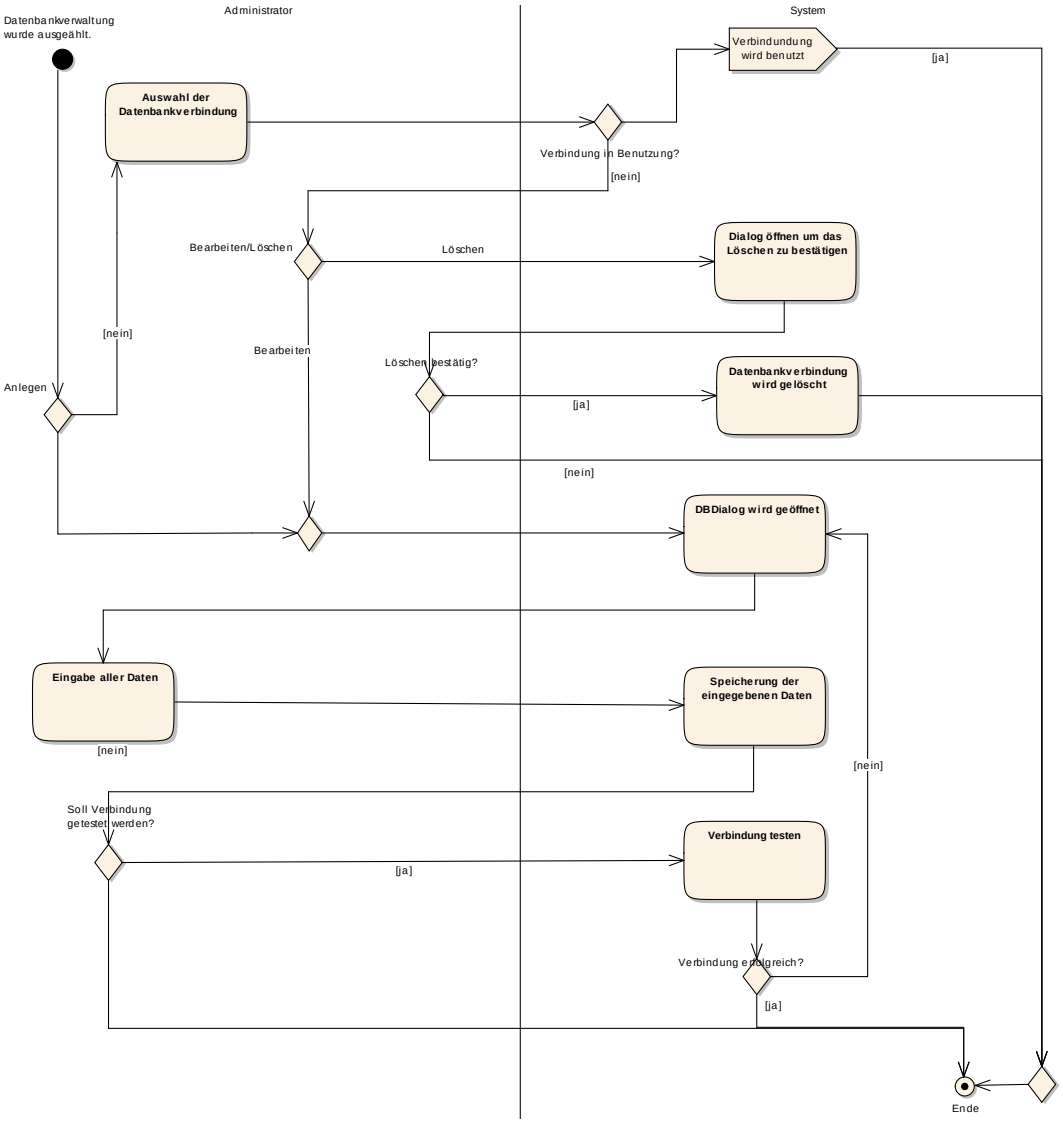

Abbildung C.5: Das Aktivitätsdiagramm Datenbankverbindung verwalten

C.3.2 Opc-Verbindung verwalten

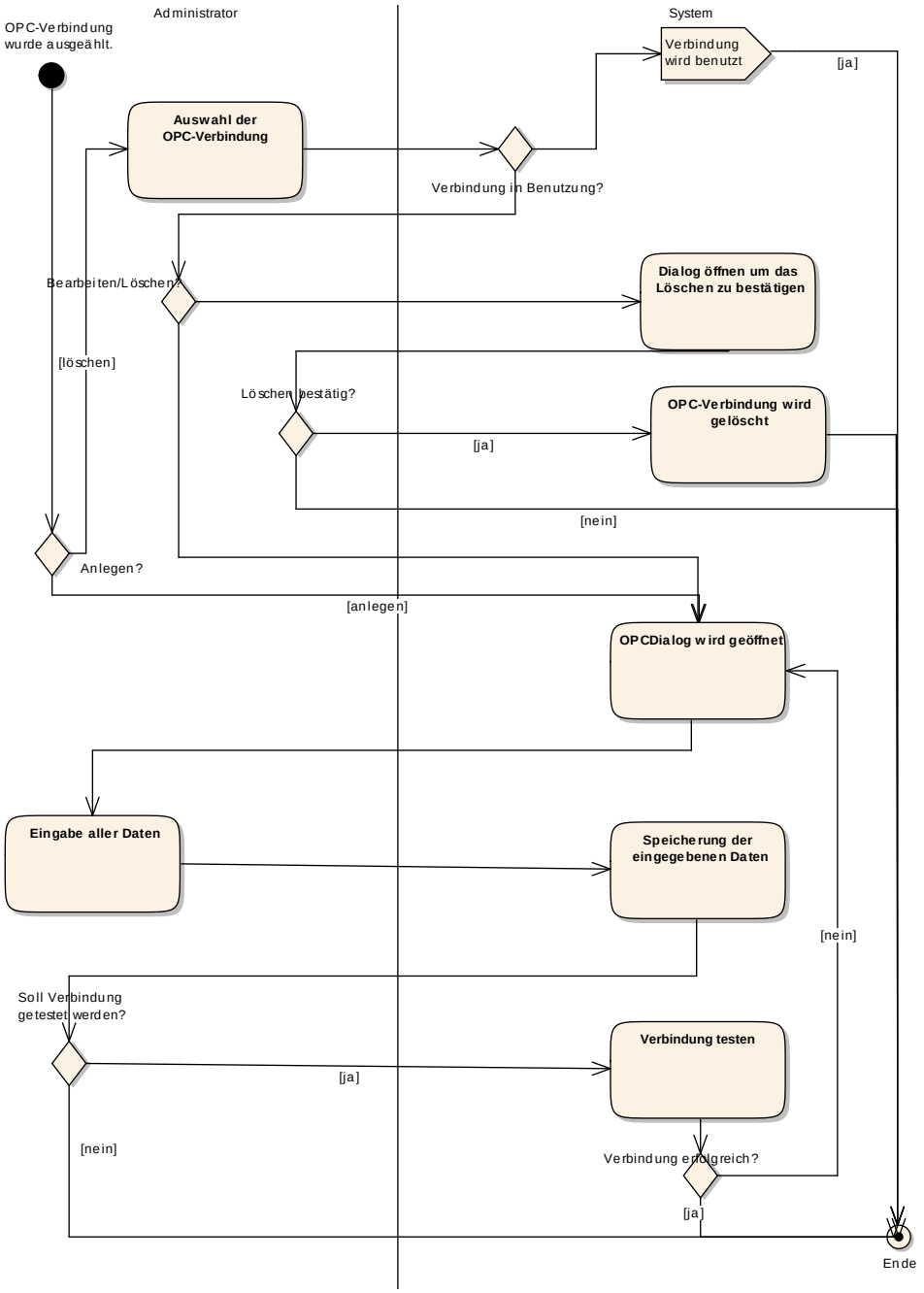

Abbildung C.6: Das Aktivitätsdiagramm Datenbankverbindung verwalten

C.3.3 Konfiguration verwalten

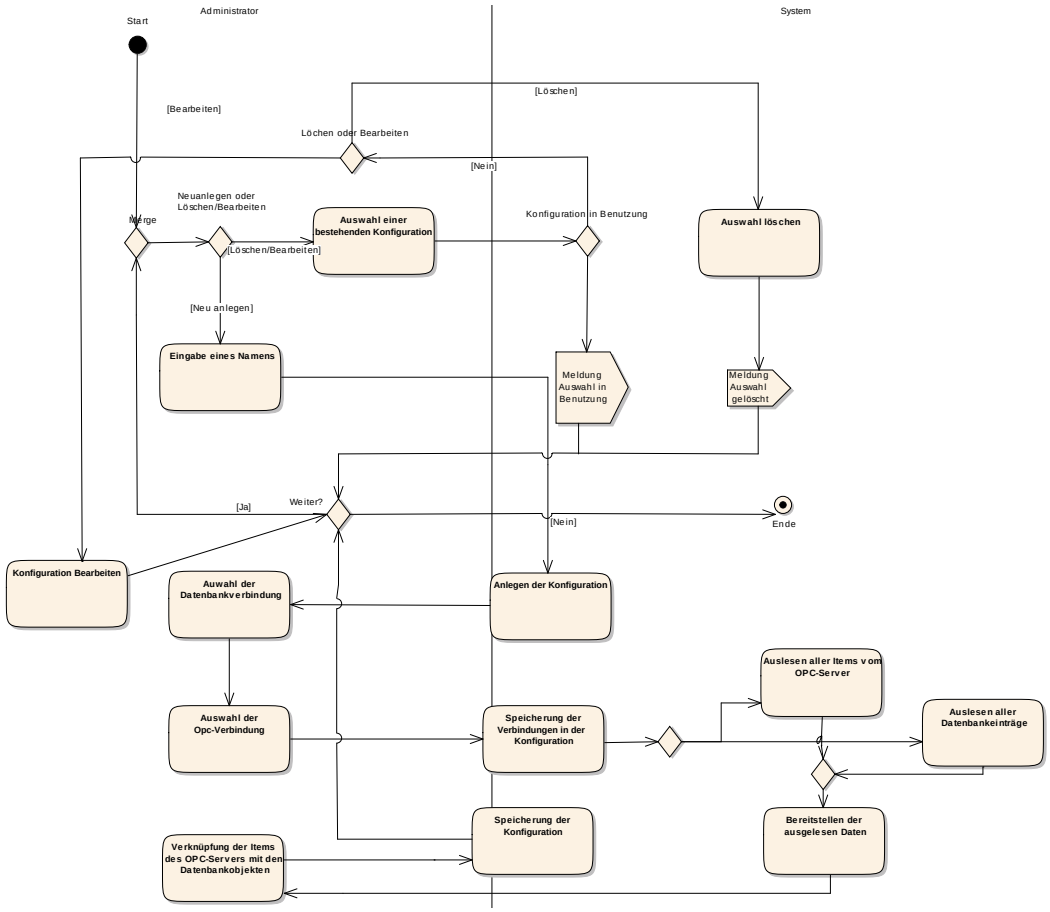

Abbildung C.7: Das Aktivitätsdiagramm Datenbankverbindung verwalten

C.3.4 Datentransfer verwalten

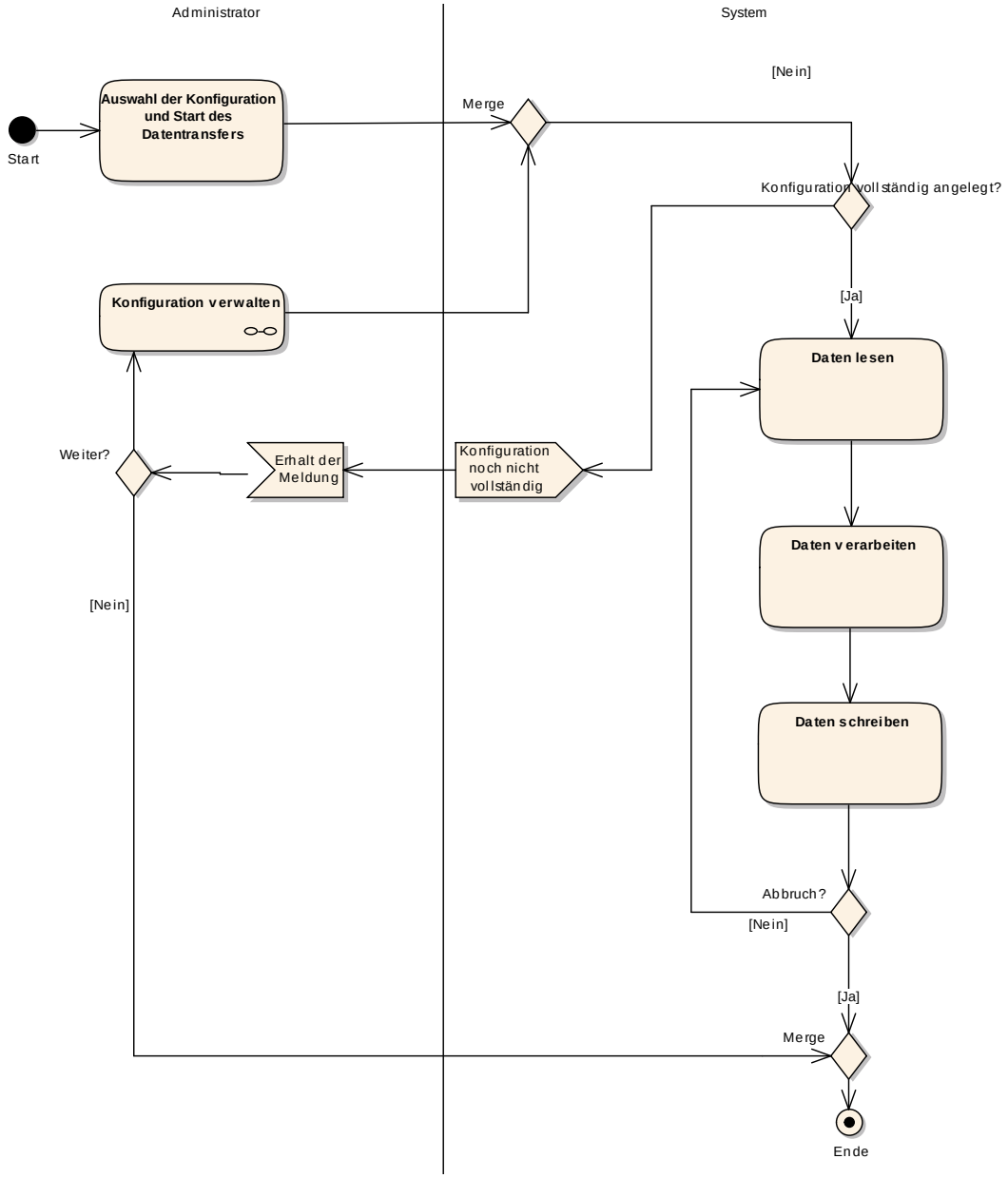

Abbildung C.8: Das Aktivitätsdiagramm Datenbankverbindung verwalten

C.4 Die Sequenzdiagramme

Das Sequenzdiagramm Konfiguration anlegen wurde unter Punkt B.5 bereits beschrieben. Um jedoch eine Konfiguration angelegen zu können, fehlen noch die Datenbank- und Opc-Verbindung anlegen sowie die Verwaltung der Items. Da sich die Schnittstelle in die zwei Teile Konfiguration und Datentransfer aufteilt, ist auch die Verwaltung des

Datentransfers mit dem Anlegen und Stoppen mit den dazugehörigen Diagrammen erfasst.

C.4.1 Datenbank-Verbindung anlegen

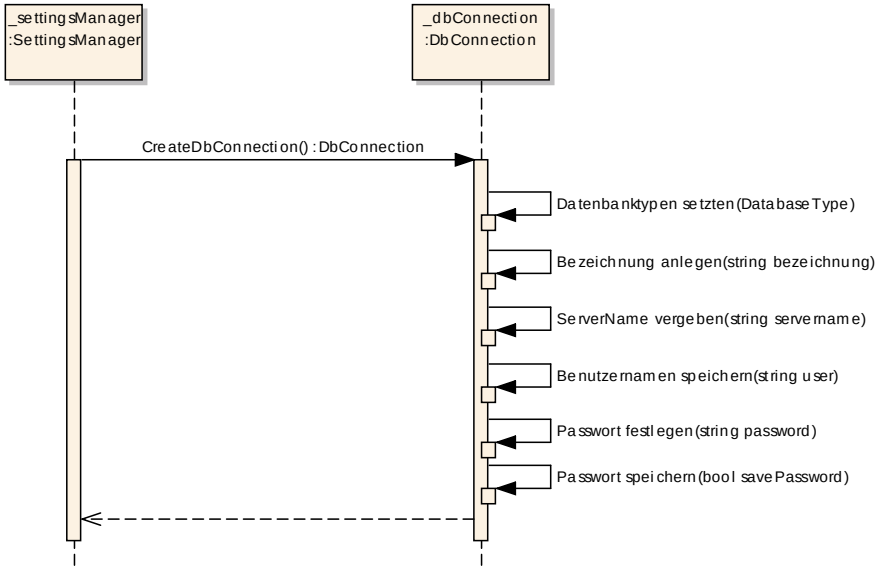

Abbildung C.9: Das Sequenzdiagramm Datenbank-Verbindung anlegen

C.4.2 Opc-Verbindung anlegen

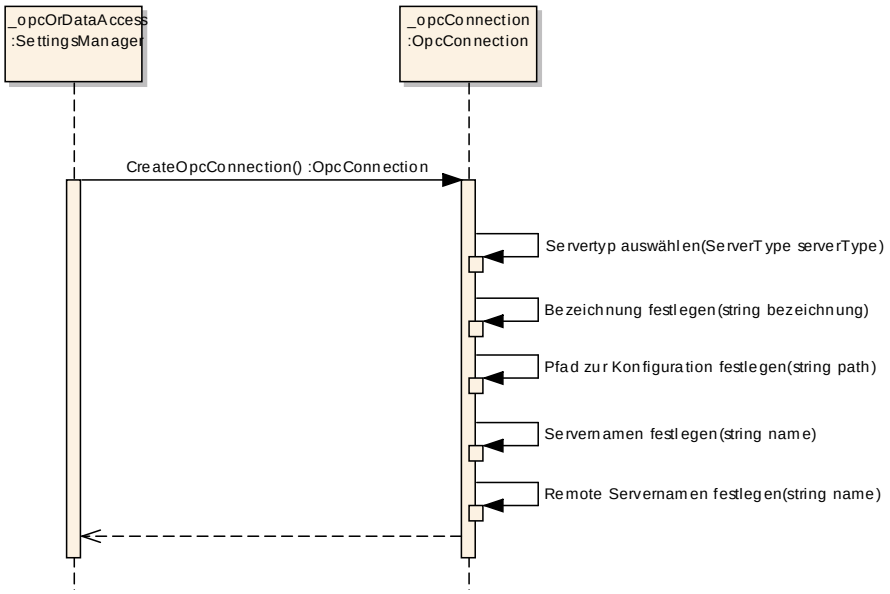

Abbildung C.10: Das Sequenzdiagramm Opc-Verbindung anlegen

C.4.3 Konfiguration verwalten

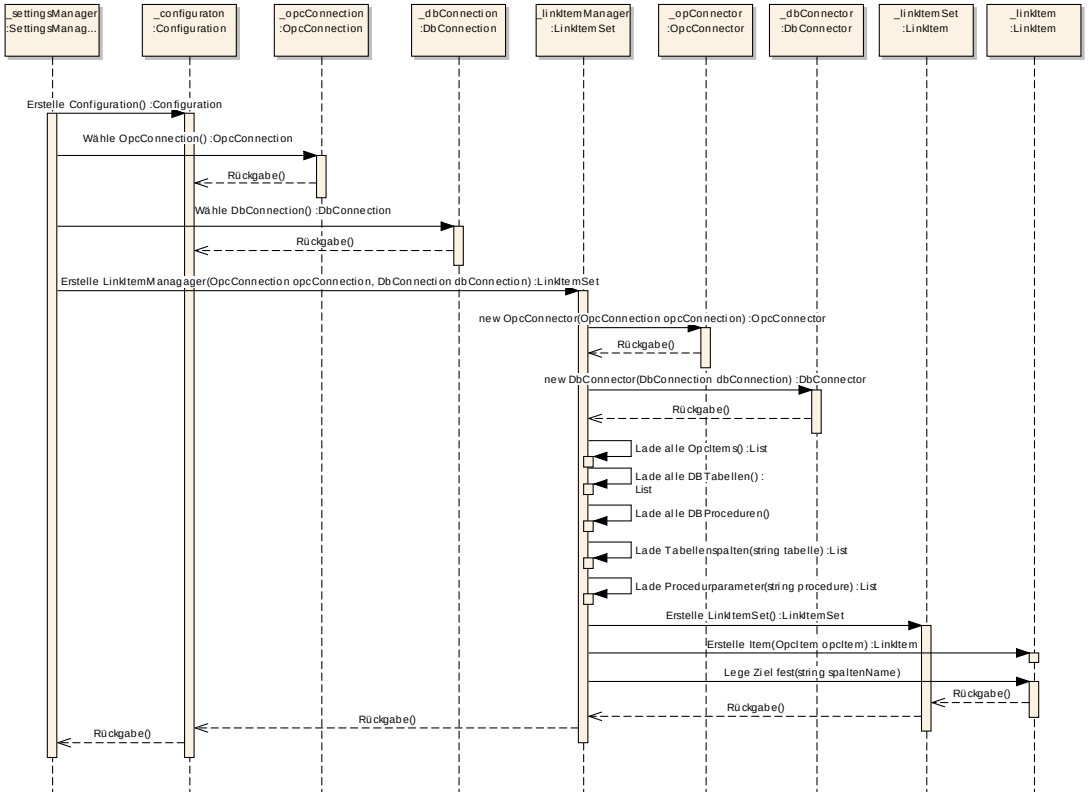

Abbildung C.11: Das Sequenzdiagramm Konfiguration anlegen

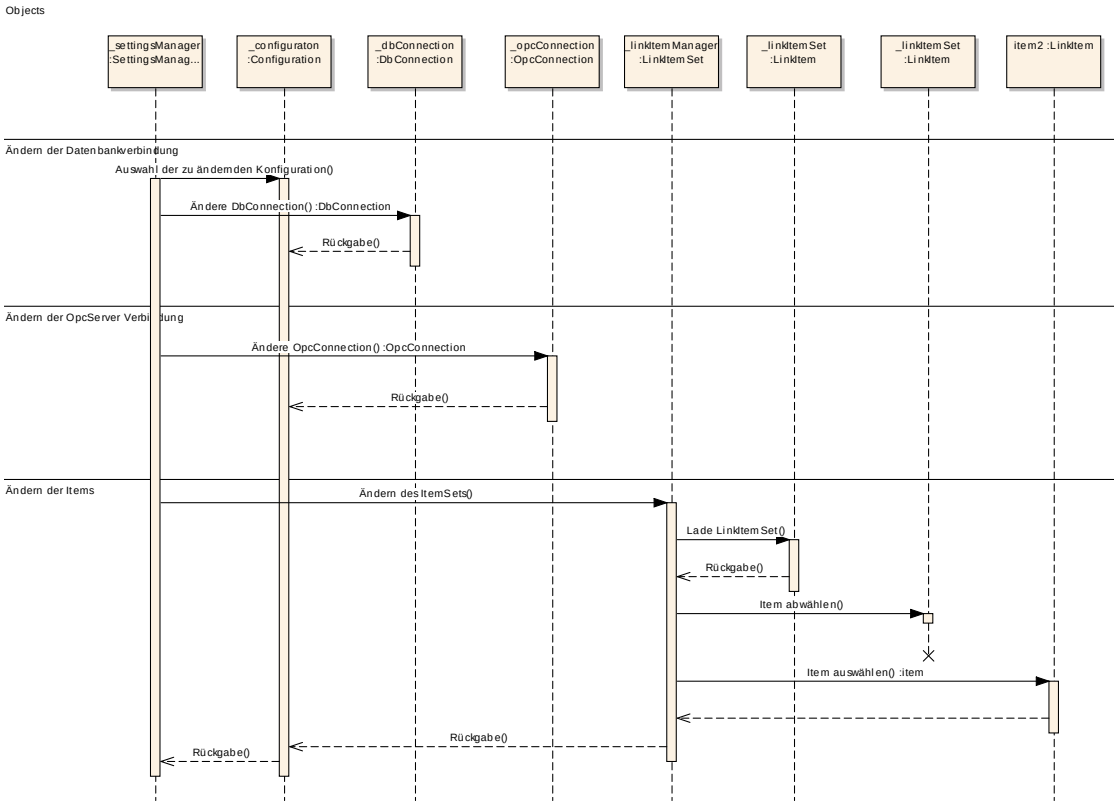

Abbildung C.12: Das Sequenzdiagramm Konfiguration ändern

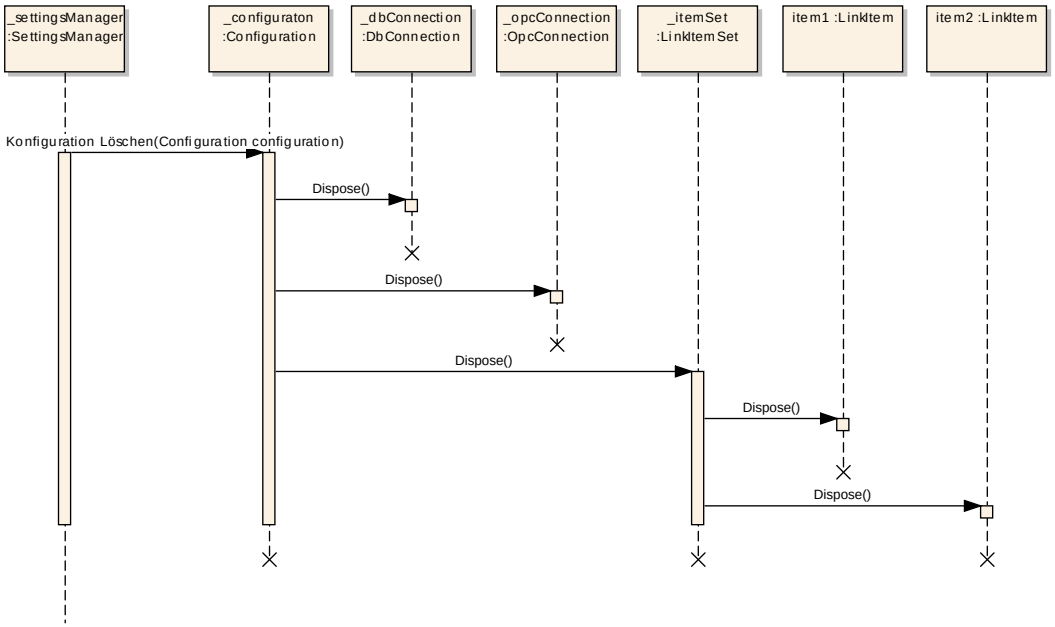

Abbildung C.13: Das Sequenzdiagramm Konfiguration löschen

C.4.4 Datentransfer verwalten

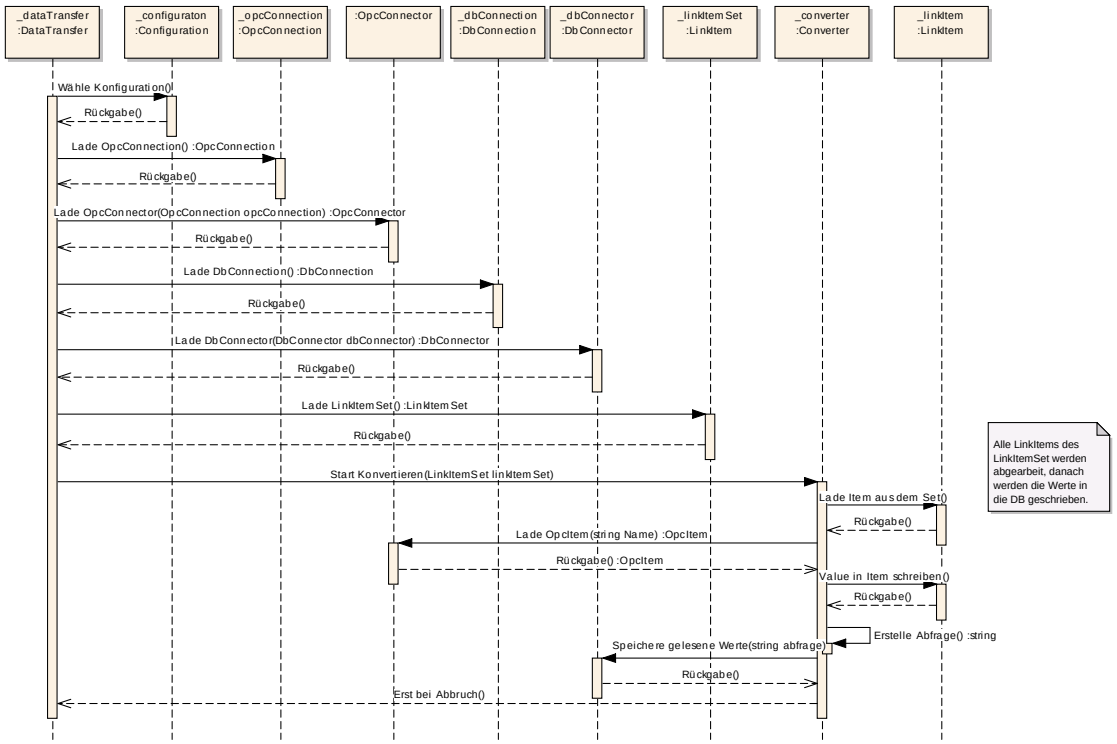

Abbildung C.14: Das Sequenzdiagramm Datentransfer starten

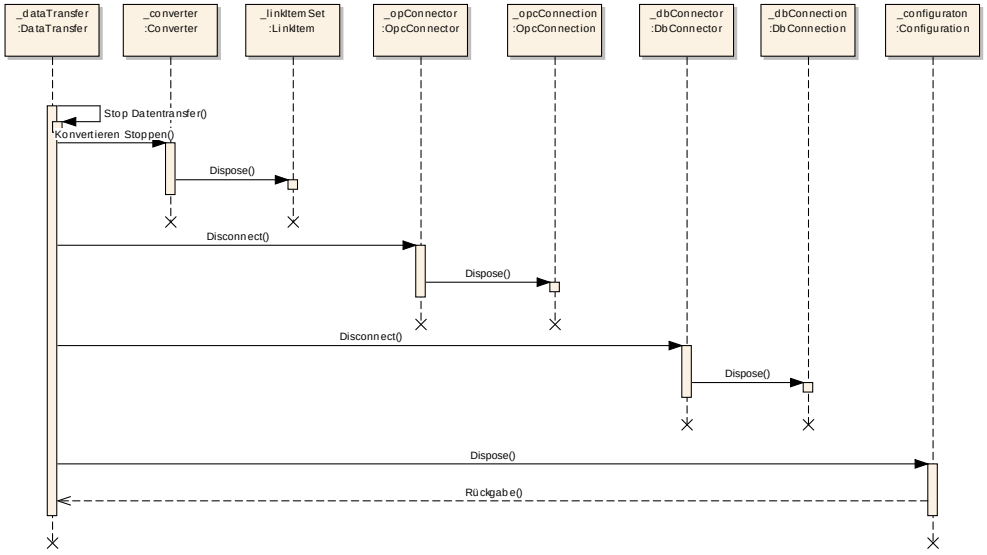

Abbildung C.15: Das Sequenzdiagramm Datentransfer stoppen

Anhang D: Das Data Dictionary

Pos	Name	Beschreibung
1	Opc-Server-Verbindung	OPCID + Servertyp + Bezeichnung + Server-Name + RemoteServer
2	OPCID	GUID
3	Servertyp	[INAT|SimaticNET]
4	Bezeichnung	string
5	ServerName	string
6	RemoteServer	string

Tabelle D.1: DataDictionary Opc-Server-Verbindung

Pos	Name	Beschreibung
1	Datenbank-Verbindung	DBID + Datenbanktyp + Bezeichnung + ConnectionString
2	DBID	GUID
3	Datenbanktyp	[ORACLE|SQLServer]
4	Bezeichnung	string
5	ConnectionString	string

Tabelle D.2: DataDictionary Datenbank-Verbindung

Pos	Name	Beschreibung
1	LinkItem	LinkItemID + DBName + Zielfunktionstyp + Zielfunktion + OpcName + TimeStamp + Value + Valuetype
2	LinkItemID	GUID
3	DBName	string
4	Zielfunktionstyp	[Tabelle|Datenbank]
5	Zielfunktion	string
6	OpcName	string
7	OpcOtemHandle	string
8	OpcSize	string
9	TimeStamp	DateTime
10	Value	object
11	ValueType	enum

Tabelle D.3: DataDictionary LinkItem

Pos	Name	Beschreibung
1	LinkItemSet	LinkItemSetID + {LinkItems}
2	LinkItemSetID	GUID
3	LinkItem	LinkItem

Tabelle D.4: DataDictionary LinkItemSet

Pos	Name	Beschreibung
1	Konfiguration	ConfigID + Bezeichnung + OpcConnection + DbConnection + LinkItemSet + SignalsValidName
2	ConfigID	GUID
3	Bezeichnung	string
4	OpcConnection	Opc-Server-Verbindung
5	DbConnection	Datenbank-Verbindung
6	LinkItemSet	LinkItemSet
7	SignalsValidName	string

Tabelle D.5: DataDictionary Konfiguration

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 02. September 2011